AUTOREFERAT

Metody zarządzania zadaniami i optymalizacja wydajności aplikacji
w hierarchicznych oraz heterogenicznych systemach obliczeniowych dużej mocy

dr inż. Krzysztof Kurowski

Poznańskie Centrum Superkomputerowo-Sieciowe
afiliowane przy Instytucie Chemii Bioorganicznej
Polskiej Akademii Nauk

Poznań, 7 maja 2018 r.

I. IMIĘ I NAZWISKO

Krzysztof Kurowski

II. POSIADANE STOPNIE I TYTUŁY NAUKOWE

1. Stopień doktora nauk technicznych - Politechnika Poznańska, Wydział Informatyki; dyscyplina - Informatyka;
   - rok nadania: 2009r.
   - tytuł rozprawy doktorskiej: "*Multicriteria Resource Management in Grid Environments with Dynamic Descriptions of Jobs and Resources*"
   - promotor: prof. dr hab. inż. Jan Węglarz

2. Tytuł zawodowy magistra - Politechnika Poznańska, Wydział Informatyki; kierunek - informatyka; specjalizacja - Inteligentne Systemy Wspomagania Decyzji;
   - rok nadania: 2001 r.
   - tytuł pracy magisterskiej: "*Multi-objective genetic local search methods for the flow shop problem*"
   - promotor: dr inż. Maciej Hapke

3. Tytuł zawodowy inżyniera - Politechnika Poznańska, Wydział Informatyki i Zarządzania; kierunek - Informatyka:
   - rok nadania: 1999 r.
   - tytuł pracy inżynierskiej: "*Predicting Job Execution Times in the Grid*"
   - promotor: dr inż. Jarosław Nabrzyski

III. DOTYCHCZASOWE ZATRUDNIENIE W JEDNOSTKACH NAUKOWYCH

1. Analityk Systemów Komputerowych
   - Pracodawca: Poznańskie Centrum Superkomputerowo-Sieciowe ICHB PAN
   - Okres: 1998 – 2005
2. Starszy Specjalista Analityk Systemów Komputerowych
   - Pracodawca: Poznańskie Centrum Superkomputerowo-Sieciowe ICHB PAN
   - Okres: 2005 - 2007
3. Senior Researcher
   - Pracodawca: University of Queensland, Australia
   - Okres: 2007 - 2008
4. Kierownik Działu Aplikacji
   - Pracodawca: Poznańskie Centrum Superkomputerowo-Sieciowe ICHB PAN
   - Okres: 2008 – 2015
5. Kierownik Pionu Zastosowań
   - Pracodawca: Poznańskie Centrum Superkomputerowo-Sieciowe ICHB PAN
   - Okres: 2015 -
6. Wizyty naukowe
   - University of Wisconsin-Madison, Stany Zjednoczone, wizyta naukowa w ramach realizacji międzynarodowego projektu badawczo-rozwojowego GridLab EU IST-2001-

32133 połączona z udziałem w konferencji naukowej, zaproszenie Prof. Miron Livny, maj 2003;

- Argonne National Laboratory, Stany Zjednoczone, wizyta naukowa w ramach realizacji międzynarodowego projektu badawczo-rozwojowego GridLab EU IST-2001-32133, zaproszenie Prof. Ian Foster, luty-marzec 2004;
- University of Southern California, Stany Zjednoczone, wizyta naukowa w ramach realizacji międzynarodowego projektu badawczo-rozwojowego GridLab EU IST-2001-32133, zaproszenie Prof. Ewa Deelman, lipiec-sierpień 2005;
- University of Melbourne i Monash University, Australia, wizyta naukowa w ramach realizacji projektu międzynarodowego QosCosGrid FP6-2005-IST-5 033883 połączona z udziałem w cyklu seminariów naukowych, zaproszenie Prof. David Abramson i Prof. Rajkumar Buyya, luty 2008;
- University of Louisiana, Stany Zjednoczone, wizyta naukowa w ramach realizacji projektu międzynarodowego niewspółfinansowanego UCoMS Nr 469/1/N-USA/2009 połączona z udziałem w konferencji naukowej, zaproszenie Prof. Edward Seidel, listopad 2010;
- University of Tokyo, Japonia, wizyta naukowa połączona z udziałem w seminariach wewnętrznych, zaproszenie Prof. Satoshi Matsuoka, marzec 2013;
- University of Notre Dame, Stany Zjednoczone, wizyta naukowa połączona z udziałem w konferencji naukowej, zaproszenie Prof. Jarosław Nabrzyski, sierpień 2011;

IV. WSKAZANIE OSIĄGNIĘCIA NAUKOWEGO WYNIKAJĄCEGO Z art. 16 ust. 2 USTAWY O STOPNIACH NAUKOWYCH I TYTULE NAUKOWYM ORAZ O STOPNIACH I TYTULE W ZAKRESIE SZTUKI (Dz. U. 2016 r. poz. 882 ze zm. w Dz. U. z 2016 r. poz. 1311.)

A. TYTUŁ OSIĄGNIĘCIA NAUKOWEGO

Metody zarządzania zadaniami i optymalizacja wydajności aplikacji w hierarchicznych oraz heterogenicznych systemach obliczeniowych dużej mocy

B. LISTA PRAC WCHODZĄCYCH W SKŁAD OSIĄGNIĘCIA NAUKOWEGO

Wyniki moich badań wchodzących w skład osiągnięcia naukowego w uporządkowanym chronologicznym zestawieniu przedstawiono poniżej. W zestawieniu uwzględniono 10 artykułów naukowych opublikowanych w wiodących czasopismach znajdujących się w bazie Journal Citation Reports (JCR). Dla każdej z prac podano wartości współczynnika wpływu (Impact Factor; IF) oraz 5-letniego współczynnika wpływu (5-year Impact Factor; 5-y IF), a także liczbę punktów z listy Ministerstwa Nauki i Szkolnictwa Wyższego (MNiSW).

[P1] S. Bąk, M. Krystek, K. Kurowski, A. Oleksiak, W. Piątek, J. Węglarz, GSSIM–A tool for distributed computing experiments, *Scientific Programming* 19 (4), 231-251, DOI: 10.3233/SPR-2011-0332, 2011

- Wydawnictwo: HINDAWI LTD; Impact Factor (IF): 0,627 ; punkty MNiSW (PM): 20;
- Cytowania: Web of Sciences - 11, Scopus - 17, Google Scholar – 30.

[P2] M. Błażewicz, SR. Brandt, M. Kierzynka, K. Kurowski, B. Ludwiczak, J. Tao, J. Węglarz, CaKernel – a parallel application programming framework for heterogenous computing architectures, *Scientific Programming* 19 (4), pp. 185-197, DOI: 10.3233/SPR-2011-0333, 2011

- Wydawnictwo: HINDAWI LTD; Impact Factor (IF): 0,627 ; punkty MNiSW (PM): 20;

- Cytowania: Web of Sciences - 13, Scopus - 13, Google Scholar – 18.

[P3] M. Ciżnicki, K. Kurowski, A. Plaza, Graphics processing unit implementation of JPEG2000 for hyperspectral image compression, *Journal of Applied Remote Sensing* 6 (1), 061507-1-061507-14, DOI: 10.1117/1.JRS.6.061507, 2012
- Wydawnictwo: SPIE-SOC PHOTO-OPTICAL; Impact Factor (IF): 1,107; punkty MNiSW (PM): 20;
- Cytowania: Web of Sciences - 4, Scopus - 14, Google Scholar – 28.

[P4] K. Kurowski, A. Oleksiak, W. Piątek, J. Węglarz, Hierarchical scheduling strategies for parallel tasks and advance reservations in grids, *Journal of Scheduling* 16 (4), 349-368, DOI: 10.1007/s10951-011-0254-9, 2013
- Wydawnictwo: SPRINGER; Impact Factor (IF): 1,281 ; punkty MNiSW (PM): 25;
- Cytowania: Web of Sciences - 10, Scopus - 16 , Google Scholar – 20.

[P5] M. Ciżnicki, M. Kierzynka, P. Kopta, K. Kurowski, P. Gepner, Benchmarking JPEG 2000 implementations on modern CPU and GPU architectures, *Journal of Computational Science* 5 (2), pp. 90-98, DOI:10.1016/j.jocs.2013.04.002, 2014
- Wydawnictwo: ELSEVIER SCIENCE ; Impact Factor (IF): 1,748; punkty MNiSW (PM): 30 ;
- Cytowania: Web of Sciences - 3, Scopus - 4, Google Scholar – 7.

[P6] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, MB. Belgacem, B. Chopard, D. Groend, P.V. Coveneyd, A.G. Hoekstra, Distributed multiscale computing with MUSCLE 2 - the multiscale coupling library and environment, *Journal of Computational Science* 5 (5), 719-731, DOI: 10.1016/j.jocs.2014.04.004, 2014
- Wydawnictwo: ELSEVIER SCIENCE ; Impact Factor (IF): 1,748; punkty MNiSW (PM): 30 ;
- Cytowania: Web of Sciences - 15, Scopus - 20, Google Scholar – 37.

[P7] M. Ciżnicki, M. Kulczewski, P. Kopta, K. Kurowski, Methods to load balance a GCR pressure solver using a stencil framework on multi-and many-core architectures, *Scientific Programming*, Volume 2015, Article ID 648752, DOI: 10.1155/2015/648752, 2015
- Wydawnictwo: HINDAWI LTD; Impact Factor (IF): 0,627 ; punkty MNiSW (PM): 20;
- Cytowania: Web of Sciences - 1, Scopus - 1, Google Scholar – 2.

[P8] T. Piontek, B. Bosak, M. Ciżnicki, P. Grabowski, D. Szejnfeld, K. Kurowski, Development of Science Gateways using QCG — Lessons learned from the deployment on large scale distributed and HPC infrastructures, *Journal of Grid Computing*, 14 (4), pp. 559-573, DOI: 10.1007/s10723-016-9384-9, 2016
- Wydawnictwo: SPRINGER; Impact Factor (IF): 2,766; punkty MNiSW (PM): 35;
- Cytowania: Web of Sciences - 1, Scopus - 1, Google Scholar – 3.

[P9] M. Ciżnicki, K. Kurowski, J. Węglarz, Energy aware scheduling model and heuristics for stencil codes on heterogeneous computing architectures, *J. Cluster Computing* (2017) 20: 2535, DOI: 10.1007/s10586-016-0686-2, 2017
- Wydawnictwo: SPRINGER ; Impact Factor (IF): 2,04 ; punkty MNiSW (PM): 30;
- Cytowania: Web of Sciences - 0, Scopus - 1, Google Scholar – 2.

[P10] M. Zimniewicz, K. Kurowski, J. Węglarz, Scheduling aspects in keyword extraction problem, *International Transactions in Operational Research* 25 (2018) 507–522, DOI: 10.1111/itor.12368, 2018
- Wydawnictwo: BLACKWELL; Impact Factor (IF): 1,01; punkty MNiSW (PM): 20;
- Cytowania: Web of Sciences - 0, Scopus - 0, Google Scholar – 1.

C. OMÓWIENIE CELU NAUKOWEGO W/W PRAC ORAZ OSIĄGNIĘTYCH WYNIKÓW WRAZ Z OMÓWIENIEM ICH EWENTUALNEGO WYKORZYSTANIA

Zaprezentowany cykl prac dotyczy specjalności informatyki związanej z systemami informatycznymi dużej mocy (ang. *High Performance Computing*) oraz metodami zarządzania i poprawy wydajności symulacji komputerowych. Główne zagadnienia tych specjalności skupiają się z jednej strony na projektowaniu i wdrażaniu wysokowydajnych systemów informatycznych w dużej skali udostepniając moc obliczeniową rzędu bilionów, a w perspektywie najbliższych lat trylionów operacji zmiennoprzecinkowych na sekundę (ang. *exascale*) [Don11][Kog13], a z drugiej na efektywnych metodach zarządzania oraz procedurach przydziału zadań do zasobów zapewniając ich poprawne i wydajne wykonanie. Ze względu na szereg ograniczeń oraz zmian technologicznych od ponad dekady obserwujemy kilka kluczowych zmian sprzętowo-programowych mających wpływ na rozwój rozpatrywanej klasy systemów informatycznych, w szczególności:

- zmiany na poziomie *procesorów* wynikające głównie z ograniczeń technologicznych, w szczególności ograniczeń wynikających ze zużycia energii oraz ograniczeń zwiększania częstotliwości i miniaturyzacji procesorów, a w konsekwencji zmiany architektury procesorów na architektury heterogeniczne łączące *wielordzeniowe układy* z *akceleratorami sprzętowymi* (najniższy poziom lokalnego systemu operacyjnego);
- dynamiczny rozwój na poziomie lokalnych systemów zarządzania zasobami obliczeniowymi, w tym wykorzystanie nowych technik wirtualizacji i kontenerów w odniesieniu do gwałtownego wzrostu zainteresowania oraz zapotrzebowania użytkowników na zdalne przetwarzanie danych w modelu usługi chmury obliczeniowej (ang. *cloud computing*) z wykorzystaniem nowych usług zarządzania zadaniami i eksperymentami obliczeniowymi;
- zwiększająca się *hierarchiczność* oraz *heterogeniczność* podstawowych typów zasobów obliczeniowych, w tym architektury procesorów oraz pamięci podręcznej;
- gwałtowny rozwój heterogenicznych zasobów obliczeniowych wraz z poprawą jakości połączeń sieci komputerowych, dedykowanych szybkich łączy komunikacyjnych i I/O umożliwiających fizyczne *łączenie oraz integracje geograficznie rozproszonych systemów obliczeniowych* w pełni funkcjonalne i zdalnie zarządzane systemy komputerowe dużej mocy;
- rozwój nowych wysokopoziomowych *hybrydowych środowisk programistycznych* i *uruchomieniowych* (ang. *meta- i hybrid programming and execution environments*) ułatwiających użytkownikom oraz twórcom aplikacji zwiększenie poziomu zrównoleglenia i skalowalności obliczeń w złożonych oraz często geograficznie rozproszonych systemach komputerowych;
- nowe wyzwania związane z różnymi problemami efektywnego zarządzania dla coraz to większych zbiorów zadań obliczeniowych oraz wolumenów danych składających się na zaawansowane eksperymenty obliczeniowe stosowane np. w fizyce wysokich energii – eksperyment CERN LHC Computing Grid (WLCG) [Cro12]

[For15], astrofizyce przy detekcji fal grawitacyjnych – Laser Interferometer Gravitational-Wave Observatory (LIGO) [Dee05][Dee15], fuzji jądrowej, czy biologii systemowej jak to zostało zademonstrowane w eksperymentach wieloskalowych MAPPER i ComPat [P6].

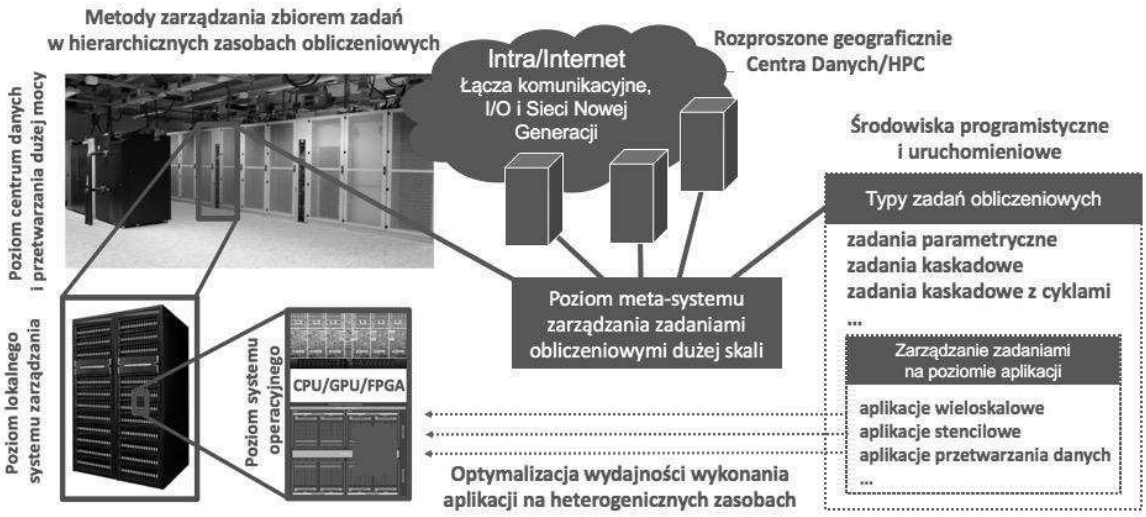
Liczne zmiany zapoczątkowane na poziomie architektury procesorów oraz pamięci podręcznej w ostatnich dziesięciu latach doprowadziły do powstania wielu nowych, stosunkowo prostych i energooszczędnych rdzeni obliczeniowych (wykorzystywanych w szerokich zastosowaniach Internetu Rzeczy – ang. *Internet of Things* i obliczeń typu ang. *edge computing*), a jednocześnie wpłynęły znacznie na dynamiczny rozwój procesorów heterogenicznych, w szczególności dedykowanych akceleratorów sprzętowych, zoptymalizowanych do wykonania określonych typów zadań. W efekcie tych zmian, lokalne systemy zarządzania zadaniami (ang. *queuing systems*) zlokalizowane w Centrach Danych/HPC wykorzystują obecnie zróżnicowane zasoby obliczeniowe od tzw. mikroserwerów opartych na energooszczędnych procesorach, poprzez klastry obliczeniowe i superkomputery ogólnego przeznaczenia, aż po wydajne układy specjalizowane do konkretnych typów obliczeń, np. stosowanych w kryptografii, analizie obrazów cyfrowych, sekwencjonowania białka, technologii blockchain, uczeniu maszynowym, itp. Ponadto, w zasadzie wszystkie wyżej wymienione wyzwania technologiczne spowodowały, nieodwracalne zmiany w systemach obliczeniowych dużej mocy, które mają charakter nie tylko ilościowy, ale też jakościowy. Odnosi się to do szeroko rozumianego procesu optymalizacji poziomu wykorzystania nowej struktury systemu informatycznego opartego na *hierarchiczności oraz heterogeniczności zasobów obliczeniowych*, który pomimo swojej złożoności powinien od strony użytkownika udostępniać spójny, łatwy w użyciu i w pełni funkcjonalny system komputerowy dużej mocy. Jest to jeden z głównych powodów podjęcia prac naukowych, których celem jest wypracowanie teoretycznych podejść oraz empirycznych testów dla nowych i kompleksowych metod rozwiązywania problemów zarządzania szeroko pojętą wydajnością obliczeń i symulacji komputerowych w odniesieniu do *różnych poziomów hierarchicznej i heterogenicznej struktury zasobów.* Jednocześnie warto zaznaczyć przyjęcie podstawowych założeń dotyczących dostępności zbiorów *wielu zadań* użytkowników w systemie ubiegających się o *współbieżny dostęp do zasobów*, jak i *optymalizacji wydajności* poszczególnych klas oraz typów *aplikacji na poziomie samych wewnętrznych struktur danych i algorytmów* w zadaniach obliczeniowych.

Warto na wstępie zaznaczyć, że kompleksowy przegląd metod zarządzania zadaniami, szeregowania oraz optymalizacji przydziału wykonania zadań na zasobach został opublikowany w literaturze naukowej [Bła07] [Węg11]. W zaprezentowanym cyklu prac oryginalny wkład dotyczy metod zarządzania zadaniami i poprawie wydajności wybranych klas aplikacji w systemach komputerowych dużej mocy z uwzględnieniem specyficznej *wielo-poziomowej struktury* systemu oraz *heterogeniczności zasobów obliczeniowych*. Od strony teoretycznych rozważań, w odniesieniu do podstawowej kategorii podzielności zasobowej, w cyklu prac rozważane są zasoby dyskretne, tj. podzielne w sposób dyskretny, a *różne horyzonty planowania* zostały podzielone na okresy, w których czas jest traktowany również jako parametr dyskretny. Wyniki badań przedstawione w cyklu publikacji, wykorzystując różne zaawansowane metody informatyczne, wzbogacają dotychczasowe prace naukowe o kompleksowe podejście do badań związanych zarówno z modelowaniem tak złożonych systemów komputerowych dużej mocy, jak i nowymi metodami zarządzania zadaniami z uwzględnieniem hierarchiczności i heterogoniczności zasobów wraz z optymalizacją szeroko pojętej wydajności systemu i aplikacji na poziomie jednego i wielu węzłów obliczeniowych. Istotnym założeniem w rozważaniach naukowych jest heterogeniczność zasobów, która ma wpływ nie tylko na stopień złożoności procesu przygotowania symulacji komputerowych w odpowiednim równoległym środowisku programistycznym i uruchomieniowym, ale jednocześnie silnie przekłada się na wydajność samych obliczeń oraz metody zarządzania

i uszeregowania wykonywanych zadań w systemie komputerowym. Z teoretycznego punktu widzenia, rozpatrywane różne kryteria jakości uszeregowania najczęściej związane są z:

- parametrami czasowymi uszeregowania zadań – *kryteria czasowe* (np. długość uszeregowania $C_{max}=max_j\{C_j\}$, gdzie kryteria czasowe są funkcjami czasów zakończenia (C_j) poszczególnych zadań);
- parametrami związanymi z wykorzystaniem dostępnych zasobów obliczeniowych – *kryteria zasobowe* (np. ważone zużycie zasobów obliczeniowych lub zużyciem energii elektrycznej w trakcie wykonywanych obliczeń na zasobie).

Jakość uszeregowania może uwzględniać jednocześnie wiele innych, specyficznych kryteriów wynikających z praktycznych uwarunkowań, które zostały uwzględnione w pracy naukowej. Specyficzne kryteria oceny jakości i wydajności mogą odnosić się do uszeregowania zbioru zadań wielu użytkowników w systemie na meta-poziomie [P4][F1] oraz rozdziału zadań na najniższym poziomie heterogenicznego zasobu, a nawet w ramach jednej aplikacji złożonej z wielu zależnych (wykonywanych współbieżnie na wielu heterogenicznych zasobach) [P9] i niezależnych zadań [P10]. Ze względów praktycznych można również rozważać jeszcze wiele innych istotnych kryteriów w odniesieniu do parametrów związanych z *kosztami*, *stabilnością obliczeń*, *odpornością na ich zakłócenia*, itp. Teoretyczny przegląd literaturowy w odniesieniu do wielu różnych zastosowań metod szeregowania dotyczący zasobów dyskretnych został przedstawiony w pracy [Bal11]. Ogólny schemat rozważanej w cyklu prac klasy systemów komputerowych dużej mocy oraz podejmowanych problemów zarządzania zadaniami uwzględniających zarówno specyficzną wielopoziomową konfiguracje zasobów obliczeniowych, jak i specyficzne wymagania zasobowe aplikacji w systemie zostały zaprezentowane poniżej na Rysunku 1.



Rysunek 1. *Poglądowy schemat zarządzania zadaniami i optymalizacja wydajności aplikacji w hierarchicznych oraz heterogenicznych systemach obliczeniowych dużej mocy obejmujący odpowiednio systemy operacyjne, lokalne systemy zarządzania zasobami oraz meta-systemy zarządzania na poziomie Centrów Danych/HPC.*

Z uwagi na możliwość praktycznego wykorzystania wyników pracy naukowej, w szczególności wdrożenia opracowanych metod zarządzania zadaniami w rzeczywistych systemach informatycznych zakładamy, że elementy składowe w rozważanym systemie w ogólności mogą obejmować zarówno zasoby obliczeniowe zarządzane na poziomie systemu operacyjnego (heterogenicznego węzła obliczeniowego), jak i zasoby zarządzane na poziomie lokalnego systemu kolejkowego, zasoby zarządzane na poziomie jednego Centrum Danych/HPC, a nawet zasoby rozproszone zarządzane na poziomie meta-systemu (ang. *meta i grid computing*) łączącego Centra Danych w różnych geograficznych lokalizacjach [P8][B15]. Poszczególne zasoby obliczeniowe, od najniższego poziomu zaczynając, mogą być zintegrowane

odpowiednio w węzły, klastry lub rozproszoną infrastrukturę obliczeniową na terenie Centrum Danych/HPC, a nawet rozproszoną geograficznie pomiędzy wieloma Centrami Danych/HPC. Należy w tym miejscu zaznaczyć, że mamy do czynienia ze strukturą systemu obliczeniowego dużej mocy, który pomimo swojej złożoności powinien zapewnić wszystkie podstawowe funkcjonalności pozwalające na jednoczesne i równoległe uruchamianie wielu różnych symulacji komputerowych (zadania wielu użytkowników rywalizują o dostęp w systemie do zasobów obliczeniowych). Ponadto, sam system powinien zagwarantować odpowiednio wysoki poziom monitorowania i dostępności wszystkich kontrolowanych prze siebie zadań i zasobów, a w sytuacjach krytycznych powinien zapewnić odpowiednie mechanizmy operacji na zadaniach, np. wstrzymanie czy zakończenie zadania oraz zabezpieczenia obliczeń i ich ponownego uruchomienia (ang. *restart and recovery*). W konsekwencji niezależnie od poziomu zlokalizowania systemu zarządzania zadaniami zakładamy, że system w dowolnej chwili (z dokładnością do przyjętych kwantów czasowych) zawsze ma pełny obraz stanu wszystkich zadań i ich wymagań zasobowych oraz aktualnego stanu zajętości zasobów. Uwzględniając w badaniach również doświadczenia z wieloma różnymi rzeczywistymi systemami komputerowymi dużej mocy przyjęte zostały następujące założenia w zaprezentowanym cyklu prac:

- w ogólności mamy do czynienia z systemem zarządza wieloma zadaniami rożnych użytkowników końcowych. Zadania trafiają do systemu informatycznego w sposób ciągły, tworząc tym samym skończony i uporządkowany *zbiór zadań* (*kolejkę zadań w systemie*) na danym poziomie, a odpowiednie metody zarządzania zadaniami z danego poziomu są wywoływane dynamicznie i najczęściej cyklicznie w celu ich przydziału do dostępnych zasobów obliczeniowych. Uruchomienie samej metody zarządzania zadaniami może nastąpić również po spełnieniu warunków określonych a priori. Dla przykładu uruchomienie metody zarządzania zadaniami może być wywołane *zdarzeniem przybycia zadania do systemu*, wywołane w określonym *okresie czasowym*, wywołane po spełnieniu danego *ograniczenia*, np. w momencie zapełnienia kolejki zadań oczekujących do określonego poziomu, itp. Samo wywołanie metod zarządzania na różnym poziomie *hierarchicznej struktury* najczęściej odbywa się asynchronicznie ze względu na przyjęte różne horyzonty czasowe planowania oraz ziarnistość parametrów opisujących zadania i zasoby w systemie. Nie zmienia to jednak innego podstawowego założenia dotyczącego możliwości komunikowania się systemów zarządzania na różnych poziomach hierarchicznej struktury systemu informatycznego, który ostatecznie powinien tworzyć logicznie spójną strukturę systemu zarządzania zadaniami jak to zostało zaprezentowane w [P4];
- dowolne *zadanie* obliczeniowe z rozpatrywanego *zbioru zadań*, jako niezależna aplikacja, może składać się z wielu *podzadań*, które mogą być wykonywane niezależnie - *zadania parametryczne* lub w ściśle określonych etapach – *zadania kaskadowe* (ang. workflow). W przypadku aplikacji modelowanej jako zadanie kaskadowe najczęściej wykorzystywana jest struktura skierowanego grafu acyklicznego (ang. *directed acyclic graph, DAG*), która definiuje wszystkie zależności i ograniczenia kolejnościowe pomiędzy zadaniami [P10]. W pracy naukowej przyjęto również rozważania bardziej zaawansowane struktury grafów cyklicznych do modelowania zaawansowanych aplikacji i wieloskalowych obliczeń [P6];
- złożone typy zadań i eksperymentów obliczeniowych wymagają uwzględnienia wymagań zasobowych oraz specyficznego wsparcia ze strony wymagań równoległych środowisk programistycznych i uruchomieniowych silnie zintegrowanych z heterogenicznymi zasobami w procesie zarządzania zadaniami. W tym przypadku, wydajność obliczeń uzależniona jest nie tyko od przydziału zadań do różnych heterogenicznych zasobów, ale też od *lokalizacji danych we/wy* oraz sposobu *dostępu do pamięci podręcznej*. Dodatkowym wyzwaniem od strony praktycznych zastosowań

jest potrzeba pełnej integracji systemu zarządzania zadaniami z wybranym środowiskiem programistycznym i uruchomieniowym, tak aby zapewnić poprawną inicjacje i koordynację uruchomienia wielu równoległych zadań (odpowiednich procesów na poziomie systemu operacyjnego), a następnie zapewnić poprawną wymianę danych i komunikatów pomiędzy zadaniami podczas ich wykonywania [P6];

- zadania mogą być wykonywane na różnych heterogenicznych zasobach, a ich wydajność podczas eksperymentów obliczeniowych powinna być mierzona i oceniana przez pryzmat *wielu kryteriów* do oceny wydajności systemu zarządzania szeregowania zadań związanych odpowiednio z *czasem wykonania zadań* [P7] i *czasów oczekiwania zadań w systemie* [B17], ale także *zużycia energii* oraz wpływu na poziom *wykorzystania zasobów obliczeniowych* [P9].

W zaprezentowanym cyklu prac rozważamy problemy należące do klasy problemów NP-trudnych [Bła07], a głównym zadaniem systemu zarządzania zadaniami jest jak najlepsze (dla przyjętych kryteriów oceny otrzymanych rozwiązań uszeregowania) wykonanie wszystkich zadań obliczeniowych przybywających do systemu komputerowego w przyjętym horyzoncie czasowym, tak aby spełnione były zarówno wszystkie wyżej wymienione zależności i ograniczenia kolejnościowe, jak i wymagania zasobowe każdego zadania w odniesieniu do *liczby procesorów* (ew. *liczby rdzeni procesora*), *rozmiaru pamięci podręcznej*, *łączy I/O*, *łączy sieciowych i komunikacyjnych*, itp. Poszukiwanie rozwiązań dla rozważanej klasy problemów nie jest trywialne i sprowadza się do zaproponowania odpowiednich *modeli matematycznych*, a następnie propozycji odpowiednich *metod heurystycznych* na różnych poziomach hierarchicznej struktury systemu komputerowego [P2][P4][P6][P7][P9][P10], których wydajność weryfikowana jest dodatkowo w *eksperymentach obliczeniowych*. Takie podejście wynika głownie z faktu złożoności obliczeniowej rozważanej klasy problemów, ale odnosi się również do wielu restrykcyjnych ograniczeń czasowych nakładanych na czas obliczeń dla procedur numerycznych implementujących metody zarządzania zadaniami w rzeczywistych systemach informatycznych. Z uwagi na przyjęte różne horyzonty czasowe planowania w systemach zarządzania zadaniami, zależnie od poziomu w hierarchicznej strukturze, tolerowane są ograniczenia czasu wykonania procedury uszeregowania zadań odpowiednio: na poziomie systemu operacyjnego mierzone w milisekundach, na poziomie systemu kolejkowego mierzone w sekundach, a na poziomie rozproszonego meta-systemu tolerowane są kwanty czasowe mierzone nawet w minutach, jak wykazano eksperymentalnie w pracy [P4]. Ponadto, na najniższym poziomie heterogenicznego węzła obliczeniowego można zaproponować *metody dokładne* zarządzania zadaniami dla wybranej klasy aplikacji i stosunkowo małych oraz testowych instancji problemów czego dowodem są eksperymentalne wyniki badań zaprezentowane w [P5][P9][Tia12]. Ze względu na przyjętą metodologię wielokryterialnej oceny jakości uszeregowań zadań na zasobach obliczeniowych w praktyce proces optymalizacji jest jeszcze bardziej złożony, albowiem poprawa wyniku na jednym kryterium oceny wydajności (np. istotnego z perspektywy użytkownika końcowego - łączny czas oczekiwania i wykonania zadań w systemie) może doprowadzić do pogorszenia wyniku na innym kryterium oceny wydajności (np. istotnego z perspektywy administratora systemu komputerowego – średni poziom obciążenia zasobów obliczeniowych), a przykłady kompleksowej wielokryterialnej analizy wyników dla rozważanych problemów zostały opisane w [P4][P10][D4][B17].

Jednym z podstawowych wyzwań dla użytkownika rozważanej klasy systemów komputerowych jest przygotowanie, a następnie efektywne uruchomienie zadania (lub zbioru zadań) w hierarchicznej i heterogenicznej architekturze zasobów obliczeniowych. Taka architektura wymaga od użytkownika znajomości a priori odpowiednich narzędzi do opisu wymagań zasobowych, szczegółowego opisu architektury wymaganego zasobu obliczeniowego oraz poprawnego określenia wszystkich zależności kolejnościowych, które należy uwzględnić na różnych poziomach i etapach przydziału, uruchomienia i monitorowania

zadania. W konsekwencji proces optymalizacji wydajności symulacji komputerowych w systemach komputerowej dużej mocy niestety nie może się odbyć bez znajomości szczegółowych założeń algorytmów i konstrukcji struktur danych w aplikacji, a jednocześnie stosownej wiedzy na temat charakterystyk mikroarchitektury sprzętowej najczęściej heterogenicznego zasobu obliczeniowego na którym aplikacja będzie uruchamiana przez system zarządzania zadaniami. Stanowi to ogromne wyzwanie dla wielu użytkowników oraz ekspertów dziedzinowych, a w praktyce najczęściej wymaga tworzenia interdyscyplinarnych zespołów łączących różne kompetencje na styku wielu różnych dziedzin oraz informatyki. Z perspektywy twórcy i użytkownika zaawansowanych symulacji komputerowych, zmieniająca się dynamicznie architektura systemów komputerowych w ostatnich latach w zasadzie uniemożliwia w sposób automatyczny zwiększanie wydajności aplikacji na zasobach obliczeniowych, w tym nawet procesorach ogólnego przeznaczenia (ang. *Central Processing Unit - CPU*). Dzieje się tak, iż od ponad dekady obserwujemy rosnącą liczbę tranzystorów i rdzeni w procesorach, co teoretycznie przekłada się na zwiększanie ich wydajności, ale niestety wymaga od użytkownika bezpośredniej ingerencji w oprogramowanie wykorzystywanego do symulacji komputerowych, w tym wielu znaczących modyfikacji mających na celu *zrównoleglenie struktur danych* i *algorytmów* na poziomie kodów źródłowych aplikacji. W konsekwencji mamy do czynienia ze złożonym procesem projektowania często nowych, równoległych oraz współbieżnych struktur i algorytmów danych z uwzględnieniem specyficznych charakterystyk architektury heterogonicznych zasobów (połączonych fizycznie oraz logicznie w hierarchiczną, ale spójną strukturę), które są wykorzystywane jako elementy składowe większych systemów komputerów dużej mocy. Z perspektywy zasobów obliczeniowych, samą wydajność procesorów w ogólności określa iloczyn częstotliwości taktowania f oraz liczby operacji (w określonej precyzji) wykonywanych w danym cyklu (ang. *Instructions Per Cycle - IPC*). Niestety, ze względu na wiele istotnych barier technologicznych oraz ograniczeń wynikających z miniaturyzacji tranzystorów oraz minimalizacji zużycia energii układów elektronicznych od wielu lat rozwój architektur mikroprocesorów skupia się nie na zwiększaniu częstotliwości taktowania procesora, lecz na zwiększaniu równoległego i współbieżnego wykonywania operacji. Zmiany architektury mikroprocesorów odbywają się na poziomie poszczególnych instrukcji (ang. *Instruction Level Parallelism – ILP*) oraz na poziomie wielowątkowości współbieżnej (ang. *Hiper-threading*) [Mar02].

W efekcie opisanych powyżej zmian, jedną z kluczowych motywacji dla zaprezentowanego cyklu prac było opracowanie holistycznego podejścia wspierającego twórców aplikacji i użytkowników w zakresie modelowania równoległych obliczeń, a następnie doboru i optymalizacji struktur danych oraz algorytmów wraz ze wsparciem na etapie projektowania eksperymentów obliczeniowych uruchamianych w rzeczywistych systemach komputerowych dużej mocy o przyjętej hierarchicznej i heterogenicznej architekturze [P2][P3][P5-9] oraz w środowisku symulatora pozwalającego na zamodelowanie najważniejszych charakterystyk zasobowych oraz różnych metod zarządzania zadaniami [P1][P4]. Ponadto, z perspektywy systemu zarządzania zadaniami, kolejnym ważnym wyzwaniem w podejmowanych pracach badawczych była potrzeba integracji i udostępnienia stosownych nowych narzędzi programistycznych wspierających użytkownika na etapie tworzenia i uruchamiania równoległych symulacji komputerowych w rozważanej klasie systemów informatycznych. W efekcie przeprowadzonych badań opracowano unikalne mechanizmy zarządzania zadaniami w systemach dużej mocy w powiązaniu z wysokopoziomowymi interfejsami programistycznymi (ang. *Application Programming Interface - API*). Otrzymane wyniki przeprowadzonych eksperymentów obliczeniowych wykazały znaczne poprawy wydajności zaawansowanych symulacji komputerowych nie tylko na etapie projektowania i tworzenia obliczeń, ale również podczas kontroli przydziału oraz poprawnego wykonania równoległych zadań na hierarchicznych i heterogenicznych zasobach z wykorzystaniem zorientowanych

dziedzinowo wysokopoziomowych interfejsów programistycznych [P2]. Jednocześnie zaproponowane nowe metody zarządzania zadaniami i dynamicznego monitorowania stanu ich wykonania oraz synchronizacji wymiany danych pomiędzy nimi pozwoliły również podnieść poziom transparentności, a tym samym ukryć przed użytkownikiem złożoność hierarchicznej, często rozproszonej [Agu11][Dee15] i heterogenicznej struktury zasobów obliczeniowych dużej mocy [P2]. Warto w tym kontekście zaznaczyć, że ogólnie przyjęte założenie dotyczące modelowania obliczeń w równoległych środowiskach programistycznych odnosi się do udostępnienia użytkownikom pełnego zbioru wysokopoziomowych interfejsów programistycznych niezbędnych do uzyskania współbieżności i równoległości obliczeń, które w postaci oprogramowania tworzy się w pewnej separacji od konkretnej mikroarchitektury heterogenicznego zasobu. Dodatkowo zakłada się, że sam proces problem *dekompozycji aplikacji* dokonywanej przez użytkownika w celu jej zrównoleglenia oraz poprawy wydajności nie jest trywialny i może odbywać się na wiele różnych sposobów, w szczególności poprzez:

- *profilowanie aplikacji* (ang. *profiling*), najczęściej z wykorzystaniem stosownych narzędzi analizy kodów, w celu wykrycia pewnych regularności w kodach źródłowych oprogramowania (ang. *data-parallel*);
- *automatyczne zrównoleglanie* podczas procesu *kompilacji* i *uruchomienia* aplikacji na heterogenicznym zasobie obliczeniowych (bez potrzeby ingerencji użytkownika w kody źródłowe oprogramowania);
- identyfikowanie procedur i modułów w kodach źródłowych aplikacji spełniających określone funkcjonalności oraz definiowaniu ich jako zbioru zadań które mogą być wykonywane na zasobach równolegle (ang. *task-parallel*).

Z perspektywy zaś komunikacji i synchronizacji obliczeń równoległych w zasadzie możemy wyróżnić dwa podstawowe modele środowisk programistycznych:

- *model przesyłania komunikatów* (ang. *Message Passing*) udostępniający użytkownikowi rozproszoną strukturę pamięci podręcznej przy założeniu, że tylko lokalna pamięć podręczna jest bezpośrednio dostępna dla zadania (procesu), a komunikacja z innymi zadaniami (procesami) odbywa się za pomocą wymiany danych w postaci komunikatów. Głównym i powszechnie przyjętym standardem w systemach komputerowych dużej mocy jest *Message Passing Interface (MPI)* – opracowany jeszcze w latach 90tych standard przesyłania komunikatów pomiędzy wieloma zadaniami (procesami), a obecnie najczęściej udostępniany i wspierany przez producentów sprzętu na wielordzeniowych procesorach ogólnego przeznaczenia [P7][B16] [Agu11] [Gro99];
- *model pamięci współdzielonej* (ang. *Shared Memory*) udostępniający pełną przestrzeń adresową pamięci podręcznej dzięki czemu, użytkownik na poziomie interfejsów programistycznych, ma możliwość łatwego skalowania rozmiaru danych niezbędnych do wykonywania obliczeń. Powszechnie przyjętym standardem jest *Open Multi-Processing (OpenMP)* - standard programowania aplikacji umożliwiający tworzenie symulacji komputerowych dla węzłów obliczeniowych z pamięcią współdzieloną [P7][Cha08]. W tym kontekście warto jeszcze wspomnieć o rozwijającym się w ostatnich latach paradygmacie programowania - *Partitioned Global Address Space (PGAS)* [Wae15]. W odróżnieniu od środowiska programistycznego OpenMP, w paradygmacie PGAS wszystkie zmienne w określonym lokalnym obszarze pamięci są prywatne dla danego wątku, a dodatkowo użytkownik może dokonać logicznego podziału pamięci współdzielonej i oznaczyć przestrzeń pamięci jako współdzieloną pamięć dla innych wątków do odczytu lub modyfikacji. W konsekwencji użytkownik może uwzględnić hierarchiczną strukturę pamięci na heterogenicznym zasobie i odpowiednio optymalizować wydajność aplikacji z uwzględnieniem lokalizacji danych [Cha07].

Niestety w ostatnich latach, głównie ze względu na dynamiczny rozwój oraz budowę różnych konfiguracji systemów dużej mocy [Don15], użytkownicy nie mają dostępu do jednolitego i zunifikowanego równoległego środowiska programistycznego i uruchomieniowego na wysokim poziomie abstrakcji programistycznej (API). Obecnie dostępnych jest bardzo wiele nowych środowisk programistycznych eksponujących z różnym stopniem skomplikowania różne unikalne cechy oraz istotne charakterystyki heterogenicznych zasobów zwiększające wydajność równoległych obliczeń. Takie dedykowane równoległe środowiska programistyczne w szczególności obserwuje się dla nowych generacji mikroprocesorów (rdzeni) stosowanych w najnowszych akceleratorach sprzętowych oraz koprocesorach instalowanych na poziomie węzła obliczeniowego jako elementu składowego systemu dużej mocy. Aktualnie możemy wyróżnić trzy wiodące środowiska programistyczne ściśle związane z heterogeniczną architekturą wysokowydajnych zasobów obliczeniowych, w tym:

- *Compute Unified Device Architecture (CUDA)* - środowisko programistyczne wysokiego poziomu oparte na języku programowania C stanowiące integralną część uniwersalnej architektury procesorów wielordzeniowych (najczęściej stosowane dla akceleratorów i kart graficznych – ang. *Graphics Processing Units (GPU)*) [P3] [P5] [P7] [P9];
- *Open Computing Language (OpenCL)* – środowisko programistyczne wysokiego poziomu oparte na języku programowania C wspierające tworzenie aplikacji działających na heterogenicznych zasobach obliczeniowych składającym się z jednostek CPU i GPU [P2] [P7];
- *OpenACC, SYCL i C++ AMP* – środowiska programistyczne rozszerzające wspomniane powyżej środowiska CUDA i OpenCL, które dzięki odpowiednim usprawnieniom w interfejsach programistycznych jeszcze bardziej ułatwiają analizę i zmianę kodów źródłowych dla równoległych obliczeń uruchamianych na heterogenicznych zasobach [Mem17].

Rozpatrywana w cyklu publikacji klasa systemów komputerowych dużej mocy, wykorzystująca w praktyce najczęściej strukturę klastrów składających się z dużej liczby heterogenicznych węzłów obliczeniowych w architekturze wielordzeniowej i akceleratorowej z rozproszoną pamięcią współdzieloną, wywarła ogromny wpływ na nowe powstanie nowych *hybrydowych środowisk programistycznych i uruchomieniowych*. W tym celu podjęto stosowne badania nad modelami uruchomieniowymi zadań i aplikacji hybrydowch, które skupiały się głównie na efektywnym połączeniu dyrektyw OpenMP (do programowania pamięci współdzielonej) z mechanizmami MPI (do oprogramowania komunikacji między węzłami obliczeniowymi w klastrze). Od strony praktycznych zastosowań oraz szerokiego wsparcia ze strony producentów procesorów nowych generacji, hybrydowe środowiska programistyczne i uruchomieniowe można z pewnym uproszczeniem sprowadzić do modelu obliczeń wykorzystujących standard *MPI + X*, gdzie *X* stanowi najczęściej jedno z wyżej wymienionych środowisk programistycznych zoptymalizowanych pod specyficzną heterogeniczną architekturę węzła obliczeniowego. Uwzględniając wymienione powyżej najbardziej istotne równoległe środowiska programistyczne dla zasobów heterogenicznych możemy w zasadzie wyróżnić dwa główne modele hybrydowe rozważane w cyklu publikacji:

- model *hybrydowego środowiska MPI + MPI*, dla którego dostosowano autorski system zarządzania zadaniami w rozproszonych środowiskach komputerowych dużej mocy – *QCG* [P8] oraz jego integracji ze środowiskiem programistycznym *OpenMPI – QCG-OMPI* wraz z kompleksową analizą wydajności obliczeń na podstawie otrzymanych wyników z eksperymentów obliczeniowych [Agu11];
- *model hybrydowego środowiska MPI + CUDA*, *MPI + OpenCL* oraz *MPI + OpenACC*, dla których wykonano szereg eksperymentów obliczeniowych w rzeczywistych systemach [P2] [P7] [B23].

Warto również zaznaczyć, że hybrydowy model oraz mechanizmy wsparcia dla komunikacji wykorzystującej wymianę i synchronizację komunikatów pomiędzy procesami doczekały się

też pełnego wsparcia ze strony popularnego języka programowania Java [Car00]. W kontekście rozważanych wymagań dla systemów komputerowych dużej mocy, a w szczególności wysokowydajnych mechanizmów współbieżnej komunikacji przekazującej komunikaty, należy nadmienić co najmniej dwa środowiska programistyczne oparte na wirtualnej maszynie Java: F-MPJ [Tab11] oraz PCJ [Now13]. W zaprezentowanych w pracach testach wskazane środowiska programistyczne umożliwiają użytkownikom uruchamianie oraz skalowanie aplikacji równoległych nawet do wielu tysięcy rdzeni procesorów. Przegląd oraz kompleksowe analizy wydajności dla współdzielonych i rozproszonych środowisk programowania z wykorzystaniem środowiska programistycznego Java przedstawiono w [Tab13].

Dotychczas w literaturze naukowej opublikowano wyniki eksperymentów obliczeniowych dla wielu różnych hybrydowych środowisk programistycznych i uruchomieniowych z uwzględnieniem specyficznych konfiguracji systemów komputerowych dużej skali w odniesieniu do popularnych algorytmów i pakietów oprogramowania wykorzystywanych w symulacjach komputerowych. Przykładowo bardzo ciekawym rozwiązaniem jest hybrydowe środowisko programistyczne i uruchomieniowe StarPU, które obsługuje planowanie zadań na wielu procesorach i procesorach graficznych [Aug12]. Innym rozwiązaniem hybrydowego środowiska programistycznego jest StarSS zapewniający ujednolicony model wykonania dla heterogenicznych zadań, który obsługuje różne algorytmy szeregowania oraz zapewnia stosunkowo proste mechanizmy równoważenia obciążenia i zarządzania danymi dla procesorów połączonych z wieloma akceleratorami graficznymi na heterogenicznym węźle obliczeniowym jak wykazano w [Pla09].

W najbliższych latach możemy spodziewać się nowych funkcjonalności oraz interfejsów programistycznych w równoległych środowiskach programistycznych i uruchomieniowych zorientowanych na optymalizację wydajności aplikacji, w którym coraz silniejszą rolę będzie odgrywać kryterium uwzględniające minimalizację kosztów związanych z przesyłaniem i dostępie do danych na poziomie samego procesora, węzła oraz klastra obliczeniowego dużej mocy [Kog13]. Sama optymalizacja dostępu i przepływu danych na heterogenicznym zasobie staje się coraz bardziej złożona ze względu na fakt, że architektury systemów komputerowych w exa-skali najprawdopodobniej nie będą w stanie zagwarantować masywnie równoległym aplikacją pełnej spójności w dostępie do pamięci podręcznej, co oczywiście będzie miało bezpośredni wpływ na zmiany modeli w równoległych środowiskach programowania uwzględniających fizyczną lokalizację danych w architekturze mikroprocesora [Pera15]. Prace naukowe podjęte w tym zakresie rozpoczęto od wyboru przykładowych dziedzinowych aplikacji wykorzystujących duże wolumeny danych, odpowiednio symulacji komputerowych stosowanych w problemach przetwarzania dużych zbiorów tekstowych [P10] oraz bioinformatyki [B20] wykonując podstawowe eksperymenty obliczeniowe w rzeczywistych środowiskach obliczeniowych. Uzyskane wyniki pozwoliły rozpocząć stosowne zmiany architektury oraz rozszerzenia funkcjonalne w środowisku symulatora [P1], a ponadto pozwoliły na zbudowanie podstawowych założeń dla modelu matematycznego i metod zarządzania zadaniami dla wybranej klasy aplikacji wraz z wstępną wielowymiarową analizą odporności rozwiązań [P10].

Cykl prac przedstawia oryginalne metody zarządzania zadaniami i techniki optymalizacji wydajności aplikacji w różnych konfiguracjach hierarchicznych oraz heterogenicznych systemów obliczeniowych dużej mocy. Przedstawiona została kompleksowa analizą parametrów kontrolnych i wielokryterialnej ocenie jakości otrzymanych rozwiązań. Zebrane metody w twórczy sposób łączą teorie szeregowania, badań operacyjnych z technikami symulacji komputerowych oraz technologiami systemów i sieci komputerowych nowych generacji zarówno w odniesieniu do modelowania w środowisku symulatora, jak i szeregu eksperymentów obliczeniowych wykonanych w rzeczywistych systemach obliczeniowych.

Omówienie głównych celów naukowych wszystkich prac wchodzących w skład prezentowanego cyklu zostanie podzielone na cztery komplementarne części, grupujące najważniejsze metody i uzyskane wyniki, do których zaliczono odpowiednio:

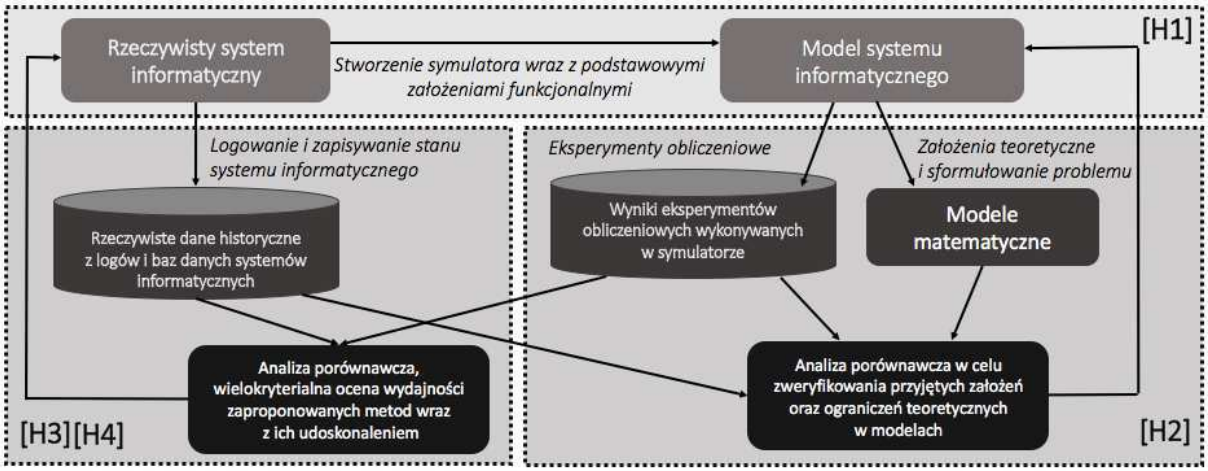
[H1] modelowanie metod zarządzania zadaniami w hierarchicznych i heterogenicznych systemach komputerowych umożliwiających kompleksowe projektowanie, rozbudowę, uruchomienie oraz odtwarzanie różnych eksperymentów obliczeniowych w środowisku symulatora **[P1]**;

[H2] modele i metody zarządzania wykonaniem zbiorem zadań na poziomie meta-systemu w hierarchicznych systemach obliczeniowej dużej mocy **[P4] [P8]**;

[H3] metody zarządzania zadaniami oraz optymalizacji wykonania aplikacji na poziomie jednego heterogenicznego węzła obliczeniowego (poziom systemu operacyjnego) **[P3]** [P5] oraz na poziomie wielu heterogenicznych węzłów obliczeniowych (poziom lokalnego systemu zarządzania zadaniami) **[P7] [P9];**

[H4] metody zarządzania złożonymi symulacjami komputerowymi oraz hybrydowymi środowiskami programistycznymi i uruchomieniowymi łączącymi hierarchiczną strukturę zarządzania zadaniami w rzeczywistych systemach obliczeniowych dużej mocy **[P2] [P6] [P10]**.

Ogólny schemat przyjętej metodyki badań w celu budowy i weryfikacji wydajności modeli oraz metod zarządzania zadaniami dla rozważanej klasy systemów informatycznych z wykorzystaniem zarówno rzeczywistych eksperymentów obliczeniowych, jak i eksperymentów obliczeniowych w środowisku symulatora, zaprezentowano na Rysunku 2. Wszystkie otrzymane wyniki eksperymentów obliczeniowych podane zostały kompleksowej i wielokryterialnej analizie porównawczej. Poszczególne obszary podjętych badań oznaczono odpowiednio [H1 – H4], wskazując jednocześnie na Rysunku 2 zakresy podjętej pracy naukowej w odniesieniu do wykorzystanych narzędzi informatycznych, eksperymentów obliczeniowych oraz stosownych analiz teoretycznych niezbędnych do budowy i weryfikacji zaproponowanych metod zarządzania zadaniami w hierarchicznych i heterogenicznych systemach dużej mocy.



Rysunek 2. *Poglądowy schemat przyjętej metodyki badań w tworzeniu i wielokryterialnej ocenie wydajności modeli i metod zarządzania zadaniami bazujących na rzeczywistych systemach komputerowych o hierarchicznych i heterogenicznych strukturach wraz ze wskazaniem podjętych obszarów prac badawczych [H1][H2][H3][H4].*

[H1] Modelowanie metod zarządzania zadaniami w hierarchicznych i heterogenicznych systemach komputerowych umożliwiających kompleksowe projektowanie, rozbudowę, uruchomienie oraz testowania różnych eksperymentów obliczeniowych w środowisku symulatora

W początkowej fazie podjętych prac badawczych nad metodami modelowania rzeczywistych systemów komputerowych, w szczególności w kontekście rozproszonych środowisk obliczeniowych typu meta-komputera (ang. grid computing), wykonano szereg niezbędnych testów i kompleksowych analiz z uwzględnieniem *rzeczywistych danych* pozyskanych z logów i historycznych baz danych zgromadzonych w systemach komputerowych dużej mocy. Prace naukowe skupione były na analizach pozwalających na pozyskanie podstawowych charakterystyk zadań i zasobów, a następnie interpretacji wykonania różnych metod zarządzania zadaniami, przydziału i wykonania zadań na zaawansowanych strukturach hierarchicznych łączących zasoby obliczeniowe. Ten ważny etap pozwolił na wypracowanie podstawowego zbioru ograniczeń i funkcjonalności niezbędnych do budowy ogólnego *matematycznego modelu systemu informatycznego* dla rozważanej klasy systemów informatycznych. Na tym etapie prac naukowych przyjęto podstawowe założenie w zaproponowanej metodologii badań, iż niezależnie od przeprowadzonych rzeczywistych eksperymentów obliczeniowych niezbędne będzie opracowanie dodatkowego, dedykowanego *środowiska symulatora* uwzględniającego wszystkie specyficzne wymagania funkcjonalne, które pozwolą na eksperymentalne zweryfikowanie i udoskonalanie opracowanych *modeli matematycznych*. Skomplikowana faza prototypowania oraz rozwoju oprogramowania symulatora zakończyła się powodzeniem, a nowe środowisko symulatora GSSIM jako dodatkowe narzędzie badawcze zostało wykorzystane w pierwszych podstawowych eksperymentach komputerowych. Pierwsze testy symulatora wykazały znaczne przyspieszenie procesu modelowania oraz oceny wydajności nowych metod zarządzania zadaniami w rozważanej klasie systemów informatycznych. W kolejnych wersjach środowisko symulatora umożliwiło implementacje i porównywanie wydajności oraz specyfiki algorytmów szeregowania zadań uwzględniających różne charakterystyki zasobowe. Możliwe stało się również implementowanie różnych algorytmów oraz reprezentacji instancji problemu i uszeregowania zadań na zasobach w różnych heurystykach bez potrzeby uruchamiania zaawansowanych i czasochłonnych eksperymentów obliczeniowych w rzeczywistych systemach komputerowych dużej mocy. Łącząc doświadczenia z eksperymentów w symulatorze oraz rzeczywistych systemach zaproponowano kompleksowe i systemowe metodologiczne podejście do rozważanych problemów badania wydajności metod zarządzania zadaniami, a ogólny zarys koncepcji przedstawiono na Rysunku 2. Dzięki przyjętej strategii przyrostowej rozbudowy symulatora o nowe funkcjonalności, referencyjne modele zarządzania i szeregowania zadaniami wraz z modułową architekturą oprogramowania symulatora GSSIM, w kolejnych etapach pracy naukowej udało się uwzględniać coraz to bardziej złożone opisy wymagań zasobowych zadań oraz struktur zasobów obliczeniowych. Rosnący poziom złożoności oraz heterogeniczność symulowanych systemów komputerowych w symulatorze GSSIM wynikał i odnosił się bezpośrednio do obserwowanych zmian technologicznych w ostatniej dekadzie, w szczególności wynikały one z dynamicznego rozwoju nowych architektur mikroprocesorów mających duży wpływ na wydajność zarówno aplikacji, jak i samych równoległych środowisk programistycznych i uruchomieniowych. Na bazie osiągniętych wyników możliwe były kolejne *udoskonalenia metod zarządzania zadaniami* oraz stosowne modyfikacje przyjętych *założeń* w symulatorze GSSIM wraz z różnymi *ograniczeniami* w *teoretycznych* i *eksperymentalnych modelach*, które w konsekwencji doprowadziły do powstania zaawansowanego środowiska symulacji komputerów dużej mocy, którego zakres funkcjonalno-użytkowy przedstawiono w artykule naukowym [P1].

Zaproponowane środowisko symulatora GSSIM umożliwiało użytkownikom wykorzystanie przykładowych konfiguracji lub stworzenie nowego modelu dowolnej fizycznej i logicznej architektury systemów komputerowych dużej mocy. Ponadto, symulator GSSIM pozwalał użytkownikom na rozbudowę istniejących oraz implementacje nowych modeli i algorytmów zarządzania zadaniami na różnych poziomach hierarchicznych struktur systemów komputerowych oraz ich wzajemne powiązania z wykorzystaniem różnych protokołów

komunikacyjnych. Wszechstronny zakres możliwości funkcjonalnych symulatora GSSIM w ocenie wydajności metod zarządzania zadaniami zaprezentowano na przykładzie różnych złożonych eksperymentów obliczeniowych w pracach [P4][B17]. Dodatkowe analizy porównawcze, bazujące zarówno na otrzymanych wynikach eksperymentów obliczeniowych w rzeczywistych systemach oraz w samym symulatorze, były podstawą do dalszych badań i usprawnień symulatora. W efekcie podjętych prac badawczo-rozwojowych, opracowano nowe moduły symulatora umożliwiając użytkownikom na jeszcze bardziej szczegółowe modelowanie wydajności zaawansowanych aplikacji i zadań uruchamianych w strukturach odzwierciedlających najnowsze architektury heterogenicznych zasobów obliczeniowych, a nowe funkcjonalności symulatora planuje się wykorzystywać w dalszej pracy naukowej.

Ze względu na duże koszty budowy oraz eksploatacji rzeczywistych systemów informatycznych, w szczególności systemów komputerowych dużej mocy, w literaturze opublikowano wiele różnych podejść do modelowania ich działania w oparciu o środowiska symulacyjne o różnym poziomie zaawansowania. Kompleksowa taksonomia w odniesieniu do narzędzi symulacyjnych została przedstawiona w [Sul04], a następnie rozszerzona o istotne charakterystyki dla rozważanej klasy systemów komputerowych dużej mocy w [P1]. Dzięki zaproponowanej modularnej architekturze środowiska symulatora GSSIM zaprojektowano, a następnie pomyślnie wdrożono wiele nowych, referencyjnych metod zarządzania i szeregowania zadań, pozwalając jednocześnie użytkownikom na dokonywanie dowolnych zmian oraz usprawnień z uwzględnieniem wielokryterialnych analiz porównawczych oraz oceny ich jakości i wydajności. Dobrym przykładem wykorzystania środowiska GSSIM w nowych obszarach praca naukowych była rozbudowa funkcjonalna symulatora zorientowanego na badanie efektywności energetycznej całych Centrów Danych/HPC – DCworms [F4]. Zakres możliwości funkcjonalnych rozbudowanego symulatora DCworms na bazie GSSIM wykracza znacznie szerzej poza ramy wybranego cyklu publikacji naukowych.

W ostatnim okresie badań związanych z optymalizacją wydajności złożonych aplikacji symulator GSSIM został udoskonalony o wiele istotnych modułów i parametrów kontrolnych, w szczególności predefiniowanych profili energetycznych i wydajnościowych dla różnych heterogenicznych zasobów na poziomie procesora, węzła i klastra obliczeniowego, co w efekcie pozwala tworzyć szczegółowe modele wykonania różnych typów aplikacji w symulatorze. W efekcie tych usprawnień, w środowisku symulatora można zamodelować, odtworzyć i porównywać osiągane wyniki do wyników licznych złożonych eksperymentów aplikacyjnych wykonywanych w rzeczywistych heterogenicznych zasobach obliczeniowych zaprezentowanych w [P2][P3][P5][P7][P9].

Workload	**Source:**	Single-user/Single-job	Single-user/Multi-job	Multi-users/Multi-job			
	Job structure:	Single-task	Dep. homogenous tasks	Indep. homogenous tasks	Dep. heteregenous tasks	Indep. heteregenous tasks	
	Job flexibility:	Rigid	Moldable	Malleable	Evolving		
	Arrival process:	Open	Closed				
	Workload composition:	Heteregenous	Same model/Homogeneus	Same model/Same structure	Same model/Diverse		
	Quality of Service:	Best effort	SLOs aware				
	Real time:	No real time	Hard real time/Aperiodic	Soft real time/Aperiodic	Hard real time/Periodic	Soft real time/Periodic	
Resources	**Heterogenity:**	Homogeneous	Heterogeneous				
	Level:	Operating system	Local system	Data Center	Meta/Grid		
	Scaling:	Static	DVFS	Outsourcing	Machine shutdown		
	Sharing:	Dedicated	Dedicated VMs	Dedicated containers	Shared VMs	OS sharing	
	Geographical coverage:	Local	Wide				
	Federation:	Federated	Single domain				
Requirements	**Scheduling criteria:**	Execution Time	Waiting Time	Resource usage	Application performance	Energy consumption	Heat dissipation
	Level:	Job-level	Task-level	Job and task-level			
	Data locality:	No affinity	Cluster affinity	Rack affinity	Node affinity	Core affinity	
	Failure model:	No failure aware	Failure aware				
	Adaptability:	Static	Adaptable				

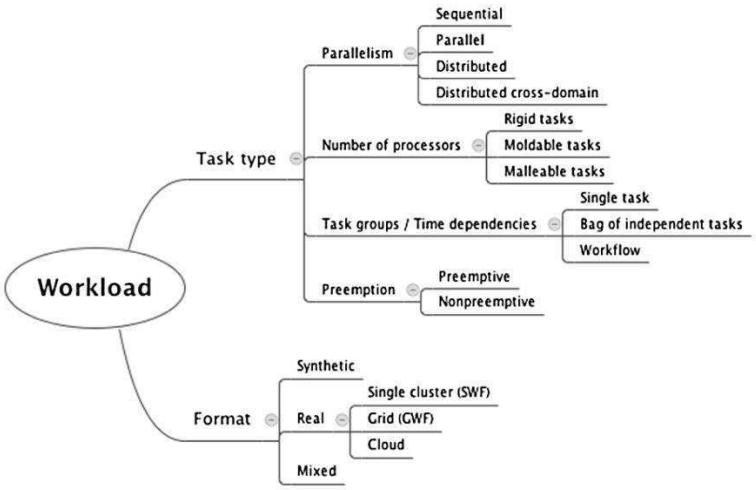
Rysunek 3. *Ogólna klasyfikacja problemów i metod zarządzania zadaniami w rozproszonych systemach komputerowych w odniesieniu do trzech podstawowych kategorii związanych z charakterystyką systemu, zadań oraz wymagań zasobowych.*

Warto również podkreślić, iż w odniesieniu do ogólnej klasyfikacji metod zarządzania oraz szeregowania zadaniami obliczeniowymi w systemach komputerowych zaproponowanej w [Lop16], opracowany i wdrożony zakres funkcjonalności symulatora GSSIM zapewnia wsparcie użytkownikowi w budowie i testowaniu różnych modeli i eksperymentów obliczeniowych z uwzględnieniem bardzo złożonych charakterystyk oraz parametrów opisujących zadania i zasoby systemów komputerowych dużej mocy. Uniwersalność i oryginalność zaproponowanych rozwiązań w symulatorze GSSIM przejawia się szczególnie w odniesieniu do możliwości łatwego modelowania hierarchicznej struktury, która w wielu praktycznych zastosowaniach składa się z wielu różnych, hierarchicznych i najczęściej zintegrowanych poziomów systemów zarządzania zadaniami. Kolejnym istotnym wkładem naukowych jest nowatorskie podejście uwzględniające wielokryterialną analizę wydajności eksperymentów obliczeniowych na podstawie zarówno rzeczywistych, jak i syntetycznych danych wykorzystywanych w eksperymentach obliczeniowych dla różnych metod i strategii zarządzania zadaniami uruchamianymi w symulatorze.
Stworzone narzędzie symulacyjne GSSIM pozwala na implementacje zaawansowanych eksperymentów obliczeniowych, badanie ich wydajności oraz optymalizacji w odniesieniu do różnych heurystycznych algorytmów stosowanych w metodach zrządzania zadaniami w dwóch podstawowych trybach. Pierwszy tryb nazywa się *off-line*, a drugi nazywa się *on-line* i dotyczy bardziej dynamicznych procesów zarządzania zadaniami wraz ze wsparciem symulowania dynamicznych procesów zarządzania równoległych i rozproszonych obliczeń dla danego wielu typów obliczeń oraz klas aplikacji. Poglądowe zestawienie trzech podstawowych kategorii związanych z charakterystyką systemu komputerowego (ang. *workload*), zasobów (ang. *resources*) oraz ich wymagań zasobowych (ang. *resource requirements*) [Lop16] w odniesieniu do możliwości oferowanych przez symulator GSSIM (szary kolor komórek w tabeli) zostało zaprezentowane powyżej na Rysunku 3. Bazując na zaproponowanej taksonomii widać wyraźnie, że aktualny zakres możliwości symulatora GSSIM pokrywa większość kluczowych kategorii oraz uwzględnia najważniejsze charakterystyki systemów rozproszonych dużej mocy.
Aktualnie realizowane prace badawczo-rozwojowe w środowisku symulatora GSSIM i jego rozbudowanej wersji DCworms skupiają się na ważnych tematach związanych z symulowaniem nowych technik wirtualizacji zasobów obliczeniowych i nowych architektur pamięci. Dodatkowo, w środowisku symulatora implementowane są zaawansowane moduły analityczne umożliwiające wielokryterialną ocenę wydajności energetycznej dla coraz bardziej złożonych aplikacji i eksperymentów obliczeniowych z uwzględnieniem również procesów dynamicznej rekonfiguracji heterogenicznych zasobów obliczeniowych.

[H2] Modele i metody zarządzania wykonaniem zbiorem zadań na poziomie meta-systemu w hierarchicznych systemach komputerowych dużej mocy

Z perspektywy problemów zarządzania zadaniami na najwyższym meta-poziomie w rozważanej klasie systemów komputerowych dużej mocy w ogólności można wyróżnić dwa podejścia: *scentralizowane* i *rozproszone*, których szczegółowe porównanie możemy znaleźć w wynikach przedstawionych w [B20]. Uwzględniając praktyczne uwarunkowania oraz opracowywane architektury systemów komputerów dużej mocy w ostatnich latach przyjęto stosowne założenia, które pozwoliły skupić podjęte prace naukowe na topologii *hierarchicznych*, *scentralizowanych* systemów komputerowych składających się co najmniej z dwóch poziomów systemów zarządzania zadaniami. Warto podkreślić, iż ziarnistość modelowania struktury systemu komputerowego w hierarchicznej topologii może znacznie się różnić i obejmować wiele różnych poziomów, w tym poziom najwyższy od heterogenicznego klastra obliczeniowego w Centrum Danych/HPC, poprzez heterogeniczne zasoby na poziomie

węzła obliczeniowego, czy jeszcze bardziej skomplikowane drobnoziarniste topologie pojedynczych heterogenicznych procesorów, rdzeni, hierarchii pamięci oraz architektury mikroprocesorów, koprocesorów czy akceleratorów graficznych. Warto również podkreślić, że w ogólności narzędzia wykorzystywane w eksperymentach symulacyjnych powinny zapewniać użytkownikowi wystarczającą elastyczność w definiowaniu modelu systemu informatycznego, a w naszym przypadku zaawansowanej architektury systemu komputerowego oraz hierarchiczności zasobów dużej mocy (w tym zarówno struktury fizycznej, jak i logicznej), czego dowodem jest funkcjonalność opisanego w poprzedniej sekcji symulatora GSSIM.
Uwzględnienie najważniejszych parametrów i charakterystyk związanych z architekturą modelowanego systemu informatycznego jest koniecznym etapem rozwoju środowiska symulacyjnego, ale jego funkcjonalność powinna również wspierać mechanizmy modelowania specyficznych zachowań oraz oddziaływania na stan różnych elementów składowych całego zamodelowanego systemu. Tym samym, jednym z najważniejszych elementów symulacji jest opracowanie *modelu obciążenia* zasobu obliczeniowego pracą niezbędną do wykonania danego zadania lub zbioru zadań. W przypadku symulatora GSSIM podstawowy model obciążenia zasobów może być wyliczany na podstawie danych historycznych opisujących duże *zbiorów zadań* pochodzących z logów *rzeczywistych systemów* obliczeniowych dużej mocy (zgodnie z ogólnie przyjętymi standardami opisu zadań [Cha99] [SWF][GWF]) lub *danych wejściowych generowane syntetycznie* na podstawie odpowiednich *parametrów* związanych z *wydajnością zadań* na poszczególnych typach zasobach obliczeniowych. Opracowany model obciążenia może oczywiście dotyczyć nie tylko jednego zadania, ale całego zbioru różnych typów zadań, które w odniesieniu do symulatora GSSIM zostały zaprezentowane na Rysunku 4.



Rysunek 4. *Klasyfikacja podstawowych typów zadań na poziomie meta-systemu zarządzania zadaniami z uwzględnieniem głównych parametrów mających wpływ na charakterystykę i obciążenie modelowanego systemu dużej mocy w symulatorze GSSIM.*

W zasadzie na dowolnym poziomie hierarchicznej struktury systemu zarządzania zadaniami modelowanie wydajności zbiorów zadań, nawet w przypadku dużych eksperymentów obliczeniowych opisanych w [P6][Cro12][For15][Dee05][Dee15], sprowadza się w pierwszej kolejności do podstawowych metod estymacji *czasu wykonania zadania* oraz *obciążenia zasobu* dla każdego rozważanego zadania szeregowanego na wybranej klasie zamodelowanych rzeczywistych heterogenicznych zasobów obliczeniowych. Dzięki wykonanym wcześniej eksperymentom obliczeniowych w środowisku symulatora GSSIM użytkownik może skorzystać z szeregu predefiniowanych podstawowych parametrów związanych z charakterystykami zasobów obliczeniowych, które przekładają się na

wydajność zadań obliczeniowych, a w szczególności na *czas wykonania zadania* uwzględniając dla przykładu:

- typ procesora oraz związaną z nim charakterystykę wydajności, np. opisywaną parametrami *Instructions Per Cycle - IPC* oraz *Instruction Level Parallelism – ILP*;
- rozmiar i dostępność pamięci podręcznej oraz hierarchii pamięci;
- oszacowany rozmiar danych we/wy potrzebnych do prawidłowego uruchomienia danego zadania na wskazanym zasobie;
- oszacowana liczba instrukcji niezbędnych do zakończenia danego zadania;
- typ i charakterystyka łącza I/O, łącza sieciowego, itp. niezbędnego do wykonania zadań zależnych/równoległych na wielu zasobach;
- inne specyficzne wymagania zadań (np. związane z konfiguracją środowiska programistycznego i uruchomieniowego).

Dla przykładu w pracach [P4] i [B17] sformułowano rożne modele oszacowania czasu wykonania p_{ij} zadania *i* z określoną liczbą instrukcji *I* na zasobie *j* bazujące na wyżej wymienionych podstawowych parametrach, w szczególności rozszerzając na wiele różnych sposobów bazowy model $p_{ij} = l_i / u_j ln(n_j)$, gdzie *u* definiuje wydajność procesora, a *n* opisuje liczbę procesorów wielordzeniowego zasobu obliczeniowego. Ponadto, w odniesieniu do hierarchicznej struktury systemu komputerowego dużej mocy, w pracy [P4] sformułowano i przebadano eksperymentalnie w środowisku symulatora GSSIM wiele referencyjnych metod zarządzania zadaniami. Uwzględniały one najczęściej stosowane parametry w metodach szeregowania zadań spotykane w rzeczywistych systemach kolejkowych odpowiedzialnych za zarządzanie systemami obliczeniowymi dużej mocy oraz podstawowych strategii szeregowania na poziomie systemu operacyjnego, szczegółowo analizowanych w [P1][B22], w tym:

- *FCFS* (ang. *First Come First Served*) – najprostsza strategia zarządzania polegająca na wyborze i szeregowaniu zadań z kolejki zadań oczekujących w kolejności w jakiej przybyły do systemu;
- *LSF* (ang. *Large Size First*) – strategia zarządzania polegająca na wyborze i szeregowaniu zadań z kolejki zadań oczekujących zgodnie z charakterystykami wymagań zasobowych określającymi rozmiar zadania (np. wymagana liczba rdzeni do uruchomienia zadania);
- *SJF* (ang. *Short Job First*) - strategia zarządzania polegająca na wyborze i szeregowaniu zadań z kolejki zadań oczekujących zadania najkrótsze;
- *PPP* (ang. *Preemtion Policy*) - strategia zarządzania polegająca na wyborze i szeregowaniu zadań z kolejki zadań oczekujących z uwzględnieniem priorytetów zadań (lub priorytety użytkowników zlecających zadania do systemu);
- *PPPD* (ang. *Preemtion Policy with Dispose)* - strategia zarządzania polegająca na porównywaniu priorytetów zadań (lub użytkowników) w kolejce zadań oczekujących z zadaniami już wykonywanymi w systemie i odpowiednim wywłaszczaniu zadań;
- *CFS* (ang. *Completely Fair Scheduling*) – strategia zarządzania polegająca na uwzględnianiu wcześniej podejmowanych decyzji przydziału zadań i wykorzystanych różnych zasobów w systemie w odniesieniu do różnych grup użytkowników;
- *RSW* (ang. *Random Selection with Weights*) – strategia zarządzania polegająca na przypisaniu zadań do kolejek zadań powiązanych z heterogenicznymi zasobami obliczeniowymi o różnej wydajności (np. szybkości procesora), gdzie prawdopodobieństwo wyboru danej kolejki powiązane jest z wydajnością danego zasobu obliczeniowego;
- *WS* (ang. *Work Stealing*) – strategia zarządzania, która podobnie jak RSW definiuje kolejki zadań dla poszczególnych zasobów obliczeniowych, a zadania przypisywane są do zasobów zgodnie z algorytmem *Round-Robin*, a w przypadku zakończenia wykonywania zbioru zadań z kolejki na danym zasobie, inne zadania z innych kolejek mogą być dodatkowo pobierane i uruchamiane;

- *HEFT* (ang. *Heterogeneous Earliest Finish Time*) – strategia zarządzania bazująca na estymacji czasów wykonania zadań w kolejce dla wszystkich dostępnych zasobów obliczeniowych, a następnie wyliczenia uszeregowania o najkrótszym czasie wykonania zbioru zadań na zasobach.

Kolejnym ważnym obszarem podjętej pracy naukowej było opracowanie modeli i przeprowadzenie szeregu eksperymentów obliczeniowych dla bardzo ważnej klasy hierarchicznych systemów obliczeniowych dużej mocy z uwzględnieniem jednocześnie *zadań wielu użytkowników* przybywających *dynamicznie* do systemu w trybie on-line, jak i zadań wymagających *rezerwacji zasobów z wyprzedzeniem* (ang. *advanced reservation*) w trybie off-line, które do tej pory nie były w literaturze naukowej poddane szczegółowej analizie [P4]. W podjętych pracach rozpatrywane były charakterystyki hierarchicznego systemu zarządzania zadaniami, które w sposób oryginalny uwzględniały jednocześnie strategie zarządzania zadaniami bez i z rezerwacją zasobów z wyprzedzeniem. Tak zaawansowane funkcjonalności dopiero w ostatnich latach zostały udostępnione użytkownikom rzeczywistych systemów komputerowych dużej mocy na bazie odpowiednio dostosowanych usług warstwy pośredniej (ang. *middleware*), np. w rzeczywistym meta-systemie zarządzania zadaniami QCG wdrażanym w rozproszonych środowiskach obliczeniowym dużej mocy w kraju i za granicą [P8]. Przez wiele lat utrudnione było zagwarantowanie wysokiego poziomu jakości oferowanej usługi (ang. *Quality of Service*) bez zastosowania odpowiednich narzędzi analitycznych dla administratorów systemów obliczeniowych dużej skali, co w efekcie najczęściej powodowało efekt odwrotny w przypadku udostępnienia mechanizmów rezerwacji zasobów z wyprzedzeniem. Innymi słowy, zamiast poprawy wydajności poprzez wykorzystanie rezerwacji zasobów z wyprzedzeniem, w wielu praktycznych zastosowaniach administratorzy obserwowali pogorszenie wydajności pracy całego systemu komputerowego dużej mocy w odniesieniu do kryteriów zasobowych (np. średni poziom wykorzystania zasobów) oraz kryteriów czasowych np. związanych z długim czasem oczekiwania lub wyższym poziomem odrzucenia zadań w systemie komputerowym dużej mocy. Szereg stosownych eksperymentów obliczeniowych w środowisku symulatora GSSIM w celu optymalizacji metod szeregowania zadań z uwzględnieniem różnych parametrów oraz charakterystyk zadań i zasobów zostało zaprezentowanych w publikacji [F1].

Istotnym obszarem pracy naukowej ze względu na praktyczne wykorzystanie wyników badań oraz potencjalne wdrożenie opracowanych nowych metod zarządzania zadaniami w rzeczywistym systemie komputerowym QCG [P8], były dedykowane eksperymenty obliczeniowe oraz kompleksowe analizy związane z odpowiednim doborem horyzontów czasowych jako parametrów kontrolnych mających wpływ na liczbę zadań w kolejkach oraz dynamiczny dobór odpowiednich strategii szeregowania zadań. Warto przypomnieć, że w przyjętych założeniach wywołanie metod zarządzania zadaniami na różnym poziomie *hierarchicznej struktury* w rozważanej klasie systemów komputerowych dużej mocy odbywa się najczęściej asynchronicznie. W konsekwencji jednym z kluczowych praktycznych problemów do rozwiązania jest dobór *optymalnych horyzontów planowania* w systemie zarządzania zadaniami z uwzględnieniem specyfiki procesu zlecania przez użytkowników zadań do systemu. Ustalenie najbardziej optymalnej konfiguracji asynchronicznego wywoływania procedur szeregowania zadań, odpowiednio na poziomie meta-systemu oraz systemów kolejkowych nie jest problemem trywialnym, jak zostało to wykazane już w [Mah99].

Zakładany okres czasu, w którym system zarządzania zadaniami zbiera informacje o aktualnym obciążeniu kontrolowanych zasobów oraz stanie dynamicznie przybywających zadań do kolejki oraz wykonywanych zadań w systemie nazywany jest *okresem planowania off-line* (pasywna fazą planowania) - δ. Jeśli $\delta = 0$ to mamy do czynienia z *okresem planowania on-line* (aktywna faza planowania), w którym procedura szeregowania wywoływana jest na dowolnym poziomie hierarchicznej struktury systemu komputerowego przez ściśle określone zdarzenie (ang. *event*), w szczególności zdarzeniem może być pojawienie się nowego zadania w systemie lub

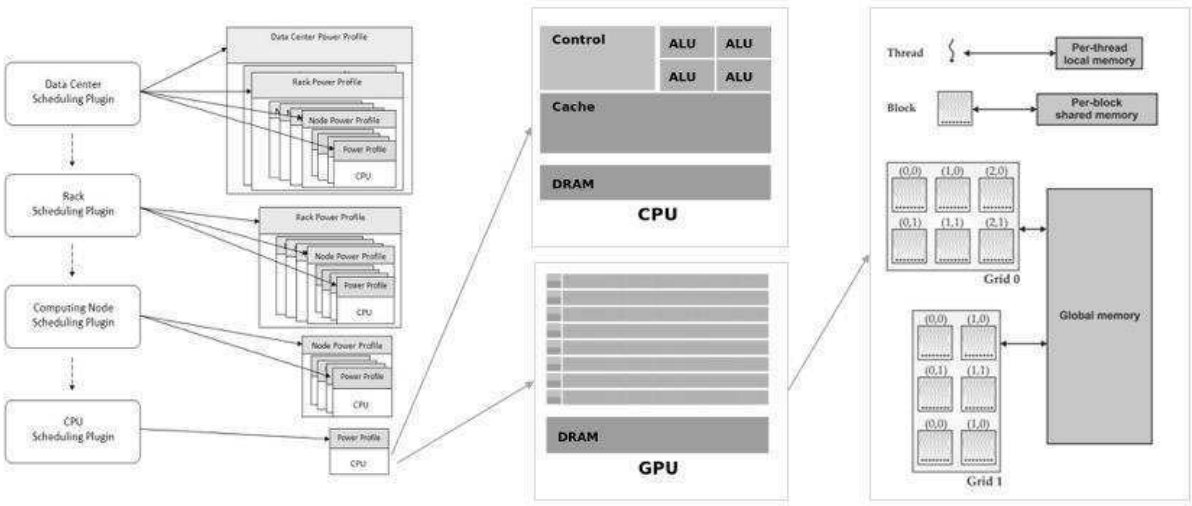
rozpoczęcie/zakończenie wykonywania zadania na zasobach obliczeniowych. W zależności od złożoności rozpatrywanych systemów obliczeniowych dużej mocy, liczby zasobów oraz zadań w hierarchicznym systemie, najczęściej rozpatruje się różne warianty $\delta > 0$, które należy jak najbardziej optymalnie określić dla każdego poziomu w hierarchicznej strukturze. Przeprowadzona i zaprezentowana w pracy [P4] kompleksowa analiza eksperymentalna wskazała wiele nowych zależności oraz istotnych parametrów kontrolnych (istotnych w szczególności dla administratorów systemów komputerowych dużej mocy), których optymalny dobór nie tylko pozwala poprawić wydajność działania systemu, ale może zapewnić lepszy poziom obsługi zadań wielu użytkowników końcowych. Inną ważną analizowaną cechą hierarchicznych systemów obliczeniowych dużej mocy, która została poddana kompleksowej analizie funkcjonalnej oraz wydajnościowej był wpływ zadań krytycznych (ang. *urgent computing*) na ogólną wydajność systemu oraz spodziewane pogorszenie poziomu jakości obsługi zbioru zadań innych użytkowników oczekujących w kolejkach zadań oraz wykonywanych na zasobach obliczeniowych, co zostało obszernie przedstawione w pracy [B17][Bec07].
W odniesieniu do rozważań teoretycznych dla podstawowych konfiguracji heterogenicznych zasobów oraz niezależnych zadań obliczeniowych obszerna analiza złożoności obliczeniowej wraz ze stosownymi modelami matematycznymi oceny wydajności metod i algorytmów szeregowania w trybie *off-line* została przedstawiona w pracach [Ble15][Mar17]. Dla zadań kaskadowych modelowanych z wykorzystaniem struktury skierowanego grafu acyklicznego (DAG) rozważania teoretyczne i różne modele matematyczne dla algorytmów szeregowania oraz wybranych heurystyk przedstawiono w pracy [Ked15]. Dla strategii szeregowania w trybie on-line teoretyczne rozważania złożoności obliczeniowej oraz przegląd modeli matematycznych omówiono w pracach naukowych dla zadań niezależnych oraz stosunkowo prostych konfiguracji zasobów obliczeniowych w [Imr03][Che15][Ama17].

[H3] Metody zarządzania oraz optymalizacji wykonania aplikacji na poziomie jednego heterogenicznego węzła obliczeniowego (poziom systemu operacyjnego) oraz na poziomie wielu heterogenicznych węzłów obliczeniowych (poziom lokalnego systemu zarządzania zadaniami)

Kolejnym istotnym obszarem podjętej pracy naukowej, komplementarnym do zaprezentowanych w poprzedniej sekcji badań w środowisku symulatora, były badania poświęcone wydajności aplikacji oraz różnych eksperymentów obliczeniowych w rzeczywistych systemach komputerowych na poziomie pojedynczych heterogenicznych węzłów obliczeniowych [P3][P5], a następnie na poziomie wielu różnych heterogenicznych węzłów obliczeniowych [P7][P9][F5][F6][B13], rozpatrywanych jako podstawowe elementy składowe w hierarchicznej strukturze systemu obliczeniowego dużej mocy. Głównym zadaniem w przyjętej metodologii badania wydajności zasobów obliczeniowych dla różnych typów zadań oraz symulacji komputerowych było pozyskanie istotnych informacji na temat wszystkich parametrów i charakterystyk niezbędnych do modelowania heterogenicznych zasobów w środowisku symulatora GSSIM na bazie eksperymentów w rzeczywistych systemach komputerowych. Otrzymane wyniki posłużyły również do budowy i weryfikacji podstawowych modeli matematycznych wykorzystywanych do estymacji czasów wykonania oraz opracowania metod zarządzania i szeregowania zadaniami, zaczynając od modelowania najniższego poziomu heterogenicznych procesorów, węzłów obliczeniowych, kończąc na najbardziej złożonych klastrach zasobów obliczeniowych dużej mocy. Warto zaznaczyć, iż dobór różnych typów zadań oraz symulacji wykorzystanych w eksperymentach obliczeniowych uwzględniał wiele kluczowych algorytmów i struktur danych wykorzystywanych w wielu różnych typów symulacji komputerowych. Obszerny przegląd wzorcowych operacji, algorytmów i struktur danych wykorzystywanych w symulacjach komputerowych wraz

z odpowiednią klasyfikacją został zaproponowany w publikacjach [Asa06] [Asa09], a następnie wielokrotnie rozbudowywany w odniesieniu do nowych typów obliczeń uwzględniających rożne charakterystyki heterogenicznych zasobów stosowanych w nowych generacjach systemów komputerowych dużej mocy [Phi11][Fen12][Str12] [Evo15].

Pierwszymi zaawansowanymi, a jednocześnie referencyjnymi eksperymentami obliczeniowymi, poddanymi kompleksowej analizie wydajności aplikacji były metody bezstratnej kompresji cyfrowego obrazu [P3], zgodnych ze standardem kodowania opisanym szczegółowo w [Tau02]. W publikacji zaprezentowane zostały podstawowe techniki optymalizacji wykonania aplikacji na poziomie jednego wielordzeniowego procesora oraz akceleratora graficznego - GPU, która wymagała stosownych modyfikacji źródeł oprogramowania w celu wydajnego zrównoleglenia algorytmów oraz struktur danych opisanych standardem JPEG2000. W efekcie tych prac osiągnięto znaczną poprawę wydajności obliczeń zarówno na poziomie całej aplikacji, jak i poszczególnych faz obliczeń. Ponadto, zaprezentowano nowy sposób oceny wydajności hierarchicznych układów procesorów wykazując, iż na pewnym poziomie abstrakcji i przy stosownych uproszczeniach można modelować i porównywać wydajność heterogenicznych architektur obliczeniowych, w szczególności układów akceleratorów graficznych. W przypadku akceleratorów graficznych GPU podstawowe charakterystyki zasobowe (możliwe do zamodelowania w rozszerzonej funkcjonalności środowiska symulatora GSSIM) najczęściej odnoszą się do znacznie większej liczby rdzeni akceleratora, mniejszego rozmiaru pamięci podręcznej i stosunkowo szybkich lokalnych łączy komunikacyjnych i łączy I/O. Innymi słowy, typową architekturę akceleratora GPU można postrzegać jako zestaw rdzeni o ograniczonej wydajności, gdzie każdy rdzeń wykonuje często tą samą instrukcję, ale działa na wielu strumieniach danych zgodnie z klasyczną architekturą SIMD (ang. *Simple Instruction Multiple Data*) według taksonomii Flynna. Co więcej, każdy rdzeń GPU ma dostęp do lokalnej pamięci współdzielonej, a także do lokalnej pamięci podręcznej w trybie wieloprocesorowym, podczas gdy procesor ogólnego przeznaczenia ma dostęp do globalnej pamięci GPU. Oznacza to, że w przypadku heterogenicznych zasobów obliczeniowych rozpatrywanych w zaprezentowanym cyklu prac najczęściej stykamy się z klasyczną architekturą systemu komputerowego NUMA (ang. *Non-Uniform Memory Access*), gdzie czas dostępu procesora/rdzenia do pamięci zależy od jej fizycznej lokalizacji na zasobie obliczeniowym, co znacznie wpływa na wydajność obliczeń dla obliczeń wymagających częstego dostępu do danych w pamięci lokalnej. Heterogeniczna architektura akceleratorów graficznych może być wykorzystana bardzo efektywnie przez różne algorytmy, które muszą wykonywać bardzo wiele powtarzalnych małych zadań (ze względu na czas wykonywania oraz zajętość zasobów pamięci podręcznej) z ograniczoną liczbą synchronizacji oraz wymiany danych. W celu zobrazowania przyjętej metodologii oceny wydajności zasobów na Rysunku 5 zaprezentowano poglądowy schemat hierarchicznej struktury metod zarządzania w odniesieniu do przyjętych metod modelowania heterogenicznych zasobów obliczeniowych CPU i GPU w architekturze NUMA zamodelowanych w środowisku symulatora GSSIM.



Rysunek 5. *Schemat hierarchicznej struktury algorytmów zarządzania i szeregowania zadaniami w odniesieniu do charakterystyki heterogenicznych zasobów CPU i GPU oraz różnych poziomów dostępu do pamięci (architektura NUMA) zamodelowanych w środowisku symulatora GSSIM.*

W przeprowadzonych eksperymentach w rzeczywistych systemach w pierwszej kolejności wykorzystano architektury heterogenicznego węzła składającego się z wielordzeniowego procesora CPU oraz akceleratora graficznego GPU demonstrując znaczną poprawę wydajności algorytmów i optymalizacje struktur danych poprzez odpowiedni podział obliczeń na tak zwane *jądra* (ang. *kernels*), które definiują minimalne jednostki obliczeń wykonanych na poszczególnych rdzeniach CPU i GPU [P3]. W ten sposób można zapewnić pełną równoległość obliczeń wykonywanych na *blokach danych*, przy czym same jądra mogą być uruchamiane jednocześnie bez potrzeby jakiejkolwiek synchronizacji. Ponadto, jądra mogą wykonywać rodzaj przetwarzania wsadowego ułożonego w postaci siatki bloków, w których każdy blok składa się z grupy wątków mających wydajny dostęp do danych poprzez współdzieloną pamięć lokalną, a ich wykonanie jest synchronizowane w dostępie do pamięci głównej. Z reguły istnieje maksymalna liczba wątków, które może zawierać dany blok. Z perspektywy zarządzania zadaniami warto podkreślić, że kilka bloków wykonywanych przez to samo jądro może być zarządzanych jednocześnie, kosztem zmniejszenia współpracy między wątkami, ponieważ wątki w różnych blokach tej samej siatki wielokrotnie nie mogą być synchronizowane z innymi wątkami, tak jak poglądowo zostało to zaprezentowane na Rysunku 5.

Na podstawie analiz wyników eksperymentów zaprezentowanych w [P3] podjęte zostały dalsze prace badawcze, a w konsekwencji udało się opracować referencyjne zasady optymalizacji, które znacznie poprawiają wydajność aplikacji na najnowszych heterogenicznych i wielordzeniowych architekturach procesorów. Sformułowano nowe algorytmy, a następnie wykazano eksperymentalnie jak ich wykorzystanie wpływa efektywnie na poprawę wydajności wybranej referencyjnie klasy aplikacji uruchamianych na pojedynczych heterogenicznych węzłach obliczeniowych w pracy [P5]. W autorskim podejściu wykorzystano metody optymalizacji dostępu do pamięci lokalnej i dedykowanych rejestrów, minimalizację liczby rozbieżnych wątków oraz zarządzanie współbieżnymi zadaniami operacji dostępu do pamięci i obliczeń zarówno na układach wielordzeniowych, jak i akceleratorze graficznym. Odpowiedni dobór referencyjnych algorytmów i struktur danych stosowanych powszechnie w standardzie kompresji cyfrowego obrazu pozwolił na kompleksową analizę różnych faz obliczeń z wykorzystaniem wzorcowych algorytmów, w których optymalizacja kodu źródłowego w znaczny sposób poprawiła wydajność obliczeń na zintegrowanym heterogenicznym węźle obliczeniowym łączącym wielordzeniowy procesor oraz akcelerator graficzny.

Następnym etapem badań były tematy związane z analizą wydajności aplikacji oraz próbą opracowania metod optymalnej dekompozycji, podziału i zarządzania zadaniami nie na jednym, lecz na wielu heterogenicznych zasobach obliczeniowych dla wybranej reprezentatywnej klasy obliczeń. Ze względu na duże zainteresowanie w literaturze klasą obliczeń modelowanych na siatkach strukturalnych – *zadania stencilowe* (ang. *stencil computing*). Obliczenia modelowane jako zadania stencilowe stosuje się dla przykładu w modelowaniu i symulacjach dynamiki płynów, modelowania geometrycznego, rozwiązywania równań różniczkowych cząstkowych lub przetwarzania obrazu i wideo. Dla tej klasy aplikacji wykonano szereg eksperymentów obliczeniowych w rzeczywistych systemach, w tym dokonano oszacowania kosztu obliczeniowego oraz skalowalności zaproponowanych algorytmów dekompozycji, podziału i zarządzania zadaniami stencilowymi na heterogenicznych zasobach obliczeniowych [P7]. Warto zaznaczyć, że zadania stencilowe były wielokrotnie traktowane jako referencyjny typ obliczeń w ocenie wydajności systemów komputerowych dużej mocy (ang. *benchmark*), a jednocześnie na przestrzeni ostatnich lat były przedmiotem zaawansowanych prac związanych z optymalizacją ich wydajności na nowych architekturach zasobów komputerowych [Sel04][Fri05][Ngu10][Tre11][Now13].
W przypadku dużych instancji problemów dla zadań stencilowych uruchamianych w rzeczywistych systemach komputerowych dużej mocy należy uwzględnić specyfikę opisywanych hybrydowych środowisk programistycznych i uruchomieniowych, które pozwalają użytkownikowi na odpowiednią dekompozycję problemu oraz współbieżne wykonywanie obliczeń w dużej skali. W związku z tym, zaprezentowano eksperymentalnie kompleksową analizę porównawczą różnych metod dekompozycji problemu dla rozważanej klasy obliczeń zadań stencilowych w pracy [P7] z wykorzystaniem odpowiednio:

- metody zarządzania zadaniami bazującej na strategii dekompozycji problemu w środowisku *MPI + X* z jednolitą partycją, w której każdy pojedynczy rdzeń odwzorowuje proces MPI bez wykorzystania pamięci współdzielonej na węźle obliczeniowym;
- metody zarządzania zadaniami bazującej na strategii dekompozycji problemu w środowisku *MPI + X* z uwzględnieniem specyficznej architektury NUMA węzła obliczeniowego oraz stosownego podziału operacji na danych pomiędzy rdzenie procesora głównego CPU oraz rdzenie akceleratora graficznego GPU [Xue14];
- metody zarządzania zadaniami bazującej na strategii dekompozycji problemu w środowisku *MPI + X* z uwzględnieniem specyficznej architektury NUMA węzła obliczeniowego, w której rdzenie procesora głównego tylko koordynują i nadzorują, a same obliczenia wykonywane są na rdzeniach akceleratorów graficznych.

Jednym z podstawowych testów wydajnościowych dla systemów komputerowych dużej mocy składających się z wielu (najczęściej w formie zintegrowanego klastra) heterogenicznych węzłów obliczeniowych dla środowiska MPI w tzw. testowanie trybie ping-pong, czyli symetrycznej wymianie komunikatów o określonym rozmiarze danych. Ten stosunkowo prosty test wydajościowy pozwala w warstwie aplikacyjnej wykonać wszystkie podstawowe pomiary związane z trwałym opóźnieniem i szybkością dostępnych łączy komunikacyjnych pomiędzy wszystkimi węzłami w rzeczywistym systemie komputerowym, a w konsekwencji pozwala oszacować podstawową *macierz kosztów*. Na bazie otrzymanej macierzy kosztów można obliczyć *minimalną ścieżkę* oraz *optymalną topologię rozmieszczenia zadań* w odwzorowaniu na heterogeniczne zasoby obliczeniowe, np. wykorzystując metodę Hilberta do obliczenia przestrzennej lokalizacji [Kam99]. Inne typowe strategie stosowane do rozmieszczania zadań na wielu heterogenicznych węzłach obliczeniowych w architekturze NUMA, które były również przedmiotem analiz w [P7], to MINALL: minimalna ścieżka między wszystkimi węzłami NUMA; MAX-ALL: maksymalna ścieżka między wszystkimi węzłami NUMA; COMPACT: minimalna ścieżka między przydzielonymi węzłami NUMA oraz BALANCED: umieszczenie zrównoważone na wszystkich dostępnych węzłach NUMA.

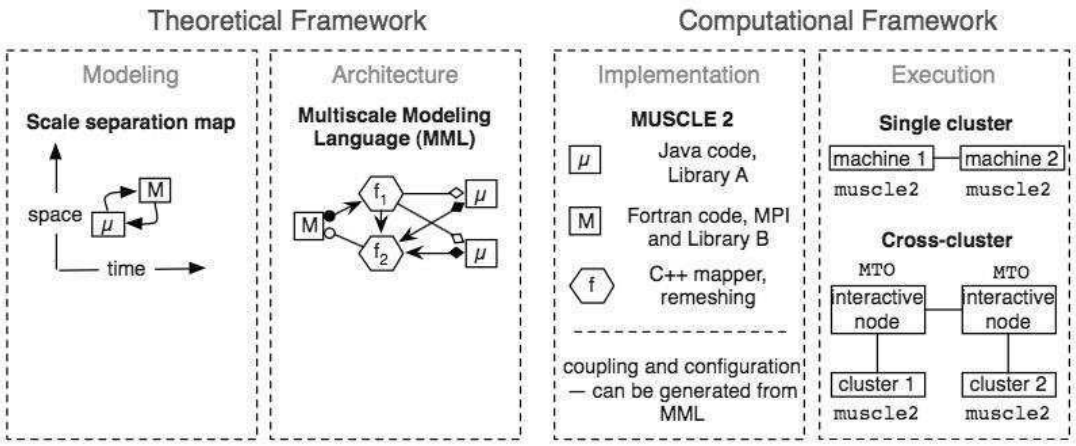
Podjęte prace badawcze pozwoliły ostatecznie na zaproponowanie w [P7] nowej elastycznej strategii dekompozycji problemu dla zadań stencilowych, która, jak wykazały testy, znacznie bardziej poprawia wydajność obliczeń dla różnych konfiguracji heterogenicznych zasobów. Zaproponowana metoda rozszerza funkcjonalność obliczania macierzy kosztów oraz wyznaczania optymalnej topologii rozmieszczania zadań o architektury NUMA, biorąc pod uwagę właściwe rozmieszczenie wątków *MPI + OpenMP/CUDA* w systemie komputerowym dużej mocy. Opracowana metoda jest w stanie skutecznie rozłożyć domenę problemu na klastry oparte wyłącznie na procesorze, w tym na maszynach NUMA, architekturach z globalną pamięcią współdzieloną oraz na klastrach tylko GPU z tzw. grubymi węzłami zawierającymi jeden procesor graficzny GPU na każdy rdzeń pojedynczego procesora GPU. Sam mechanizm partycjonowania jest wykorzystywany przed kompilacją kodu źródłowego aplikacji dla architektury docelowej, a otrzymana dekompozycja problemu wraz z przydziałem zadań do procesorów/rdzeni jest statyczna podczas wykonywanych obliczeń.
Ostatnim etapem prac badawczych z tego obszaru, było opracowanie i analiza modeli wydajnościowych oraz energetycznych dla wybranej klasy zadań stencilowych w celu zbadania zależności między algorytmami zarządzania zadaniami, a ograniczeniami energetycznymi w oparciu o zaproponowane modele matematyczne. W tym celu opracowano i przetestowano różne metody heurystyczne do rozwiązania problemu optymalnej lokalizacji zadań stencilowych na heterogenicznych zasobach w [P9]. W pracy zweryfikowano eksperymentalnie wydajność różnych metod zarządzania zadaniami, wykazując jednocześnie zależności pomiędzy wydajnością aplikacji stencilowych w odniesieniu do równoważenia obciążenia pomiędzy heterogenicznymi zasobami przetwarzającymi dane wejściowe do obliczeń. Analizy otrzymanych wyników w rzeczywistych systemach komputerowych wskazały na różne zależności oraz znaczący wpływ charakterystyk heterogenicznych zasobów i zastosowanych metod partycjonowania na jakość i wydajność obliczeń stencilowych.

[H4] Metody zarządzania złożonymi symulacjami komputerowymi oraz hybrydowymi środowiskami programistycznymi i uruchomieniowymi łączącymi hierarchiczną strukturę zarządzania zadaniami w rzeczywistych systemach obliczeniowych dużej mocy

Przedstawione w poprzednich sekcjach wyniki badań w dużym stopniu pogłębiły analizę istotnych parametrów opisujących funkcjonowanie złożonych systemów zarządzania zadaniami w systemach komputerowych dużej mocy dzięki wykorzystaniu opracowanego środowiska symulatora GSSIM. Dodatkowo, wykonane symulacje zostały zweryfikowane szeregiem eksperymentów obliczeniowych w rzeczywistych systemach zarówno w odniesieniu do zbiorów zadań, jak i poszczególnych klas złożonych aplikacji uruchamianych na heterogenicznym węźle lub klastrze węzłów. Jak zostało zaznaczone już na wstępie autoreferatu, od ponad dekady obserwujemy rosnącą złożoność systemów komputerowych dużej mocy wynikającą w dużej mierze z rosnącej liczby różnych typów heterogenicznych zasobów obliczeniowych oraz hierarchicznej topologii całego systemu superkomputerowego, jak i hierarchicznego oraz niejednorodnego dostępu do pamięci podręcznej. To wszystko w znaczący i niestety niekorzystny sposób wpływa na stopień złożoności procesu projektowania, a następnie optymalizacji zrównoleglonych struktur danych i algorytmów w aplikacjach, a w konsekwencji przyczynia się często do stosunkowo niskiej wydajności aplikacji w odniesieniu do teoretycznych możliwości sprzętowych nowych komputerów dużej mocy.
W związku z powyższym, ostatni obszar badań zaprezentowany w cyklu prac dotyczył próby usprawnienia całego procesu projektowania, tworzenia oprogramowania i zarządzania złożonymi symulacjami komputerowymi dużej skali oraz hybrydowymi środowiskami programistycznymi i uruchomieniowymi łączącymi hierarchiczną strukturę zarządzania

zadaniami w rzeczywistych systemach. W pierwszej kolejności podjęto próbę opracowania, a docelowo wdrożenia systemu informatycznego zapewniającego kluczowe funkcjonalności wsparcia dla równoległego wysokopoziomowego dziedzinowego środowiska programistycznego (ang. *Domain Specific Language*). Jednym z podstawowych założeń dla takich środowisk programistycznych jest w jak największym stopniu ukrycie złożoności heterogenicznych zasobów obliczeniowych oraz eliminacja wielu niedogodności oferowanych przez hybrydowe środowiska programistyczne i uruchomieniowe w omawianym wcześniej popularnym modelu przetwarzania *MPI + X*. Jednym z kluczowych wyzwań w podjętej pracy naukowej było dostosowanie języka wysokiego poziomu zrozumiałego dla ekspertów, opisującego wybrane zjawiska i procesy z danej dziedziny nauki do języka programowania, który umożliwiałby automatyczną translacje na hybrydowe środowisko programistyczne i uruchomieniowe. Pierwszym ważnym środowiskiem wysokiego poziomu poddanym kompleksowej analizie było zorientowane dziedzinowo środowisko programistyczne CaKernel dla opisanej wcześniej klasy zadań stencilowych [P2]. Warto zaznaczyć, iż próby stworzenia wysokopoziomowych środowisk programistycznych ukrywających złożoność infrastruktury sprzętowej komputerów dużej mocy były podejmowane przez wiele zespołów naukowych w przeszłości. Dla przykładu w odniesieniu do symulacji komputerowych bazujących na schemacie obliczeń stencilowych należy wymienić środowisko programistyczne Mint [Una11] oraz Ypnos [Orc10]. Niemniej jednak zaprezentowane rozwiązania miały zasadnicze ograniczenie związane z heterogenicznymi zasobami złożonymi tylko z jednego akceleratora graficznego, co było głównym powodem rozwoju wspomnianego środowiska CaKernel umożlwiającego uruchomienie zadań stencilowych na większych i bardziej złożonych heterogenicznych architekturach. W wyniku prac badawczo-rozwojowych rozbudowano istotnie środowisko programistyczne CaKernel, które pozwala programiście oraz użytkownikowi projektować i rozwijać wysoce wydajne bloki (jądra) obliczeń równoległych w oparciu o predefiniowane wzorce. W konsekwencji uzyskano znaczącą poprawę wydajności obliczeń dzięki automatycznej konwersji bloków i współbieżnym wykorzystaniu wielu układów GPU dostępnych w testowych rzeczywistych systemach komputerowych dużej mocy [P2]. Opracowane środowisko programistyczne CaKernel używa automatycznego generowania kodu, bazując na wysoce zoptymalizowanym zestawie szablonów kodów źródłowych i uwalnia projektantów oraz twórców aplikacji od konieczności zrozumienia szczegółów skomplikowanego procesu optymalizacji algorytmów oraz struktur danych w odpowiednim hybrydowym środowisku programistycznym i uruchomieniowym, w tym przypadku *MPI + CUDA*.

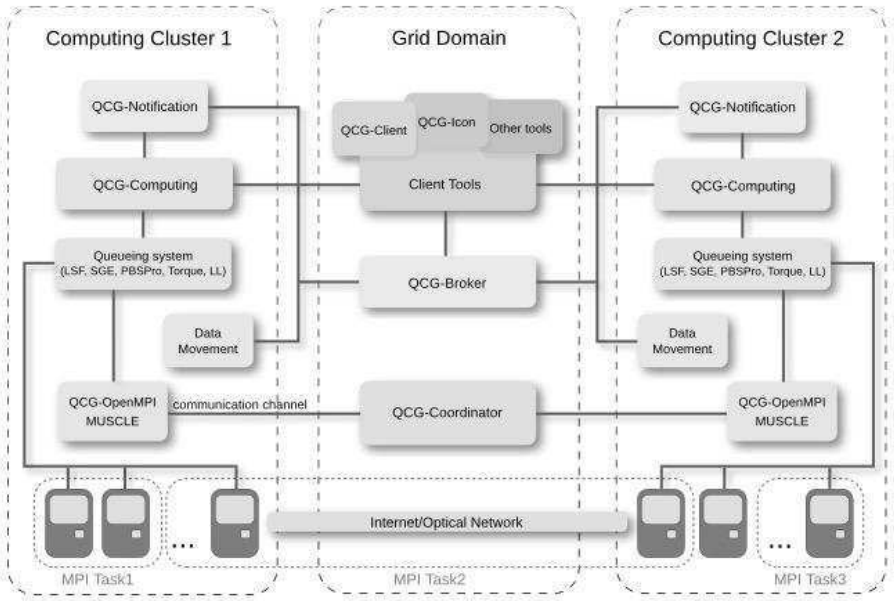
Niezwykle istotnym obszarem podjętych prac naukowych było stworzenie również innych, koncepcyjnie podobnych wysokopoziomowych narzędzi programistycznych, a następnie ich pełna integracja z systemami zarządzania zadaniami w celu zagwarantowania pełnego wsparcia dla procesu uruchamiania zadań na rozproszonych i heterogenicznych zasobach obliczeniowych. W tym celu zidentyfikowano inną bardzo ważną klasę aplikacji wielkoskalowych. W dużym uproszczeniu, zadania wieloskalowe z definicji składają się z dwóch lub większej liczby symulacji komputerowych uruchamianych w systemie komputerowym dużej mocy synchronicznie, a w praktyce każda aplikacja może być bardzo złożonym równoległym zadaniem obliczeniowy, zadaniem kaskadowym, zadaniem kaskadowym z cyklami, itp. Ogólny schemat zaproponowanej metodologii tworzenia zaawansowanych symulacji wieloskalowych od etapu analizy modelowanego rzeczywistego zjawiska/procesu, poprzez modelowanie zależności w wysokopoziomowym języku programowania MML (ang. *Multiscale Modeling Language*) opisanego w [B19], do etapu automatycznego generowania kodu źródłowego dla hybrydowego środowiska uruchomieniowego oraz fazy uruchomienia i zarządzania zadaniami w systemach komputerowych dużej mocy, został zaprezentowany poniżej na Rysunku 6.



Rysunek 6. *Ogólny schemat metodologii tworzenia i zarządzania zaawansowanymi symulacjami wieloskalowymi w systemach i klastrach komputerowych dużej mocy z wykorzystaniem systemu QCG.*

Z perspektywy omawianych metod zarządzania zadaniami oraz optymalizacji wydajności tak złożonych aplikacji warto podkreślić, że dzięki wdrożeniu systemu informatycznego QCG [P8] w rzeczywistych systemach komputerowych dużej mocy wykonano szereg eksperymentów obliczeniowych dla tej coraz ważniejszej klasy symulacji wieloskakowych. Ponadto, udało się pomyślnie zintegrować system QCG z nowym wysokopoziomowym środowiskiem programistycznym MUSCLE/2 dedykowanym i zoptymalizowanym dla aplikacji wieloskalowych. Ze względu na swoją modułową konfigurację środowisko programistyczne MUSCLE/2 wykorzystuje wysokopoziomowy język opisu wielkoskalowego eksperymentu obliczeniowego MML. Dzięki MML opis poszczególnych zadań obliczeniowych, różnych faz obliczeń i zależności wraz z wymaganiami zasobowymi jest kompletnie odseparowany od etapu implementacji, a następnie wykonania zadań na heterogenicznych i hierarchicznych zasobach. Co więcej zaproponowane podejście oddzielające etap modelowania symulacji wielkoskalowych umożliwia użytkownikom pełną swobodę zmian poszczególnych elementów symulacji bez konieczności modyfikowania całego eksperymentu obliczeniowego, a sam proces wykonania złożonych zbiorów zadań wchodzących w skład symulacji wielkoskalowych jest zautomatyzowany i optymalizowany przy wykorzystaniu systemu informatycznego QCG.
Wyniki eksperymentów otrzymane z rzeczywistych systemów komputerów dużej mocy potwierdziły nie tylko możliwość rozpraszania zaawansowanych i złożonych symulacji komputerowych na geograficznie rozproszonych systemach, ale wykorzystując wielopoziomowe strategie zarządzania zadaniami wspieranymi w systemie QCG możliwa jest pełna integracja takiego rozproszonego i hierarchicznego środowiska obliczeniowego w jeden spójny system [P6]. Dzięki zaawansowanym i wydajnym mechanizmom komunikacji oraz wymiany komunikatów, środowisko MUSCLE/2 zapewnia różnym modułom zadań zaimplementowanych w językach programowania Java, C, C ++, Python lub Fortran pełną integracje z różnymi technologiami stosowanymi w hybrydowych oraz równoległych środowiskach programistycznych i uruchomieniowych MPI, OpenMP i CUDA. Nakład pracy programisty i użytkownika MUSCLE/2 dla zaawansowanych zadań wieloskalowych jest niewspółmiernie niski w porównaniu do procesu pełnej optymalizacji kodu źródłowego dla wybranej heterogenicznej architektury zasobu obliczeniowego. Ponadto, przeprowadzone testy wydajnościowe wykazały znaczą przewagę wydajności obliczeń, w szczególności w odniesieniu do komunikacji opartej na plikach we/wy najczęściej stosowanych w zadaniach kaskadowych [P6]. Ogólna architektura rzeczywistego systemu informatycznego QCG w systemach komputerowych dużej mocy, który zapewnia wydajne zarządzanie zadaniami obliczeniowymi na etapie przydziału, uruchamiania i poprawnego wykonania została zaprezentowana na Rysunku 7. Ponadto, opracowany system informatyczny QCG integruje się z różnymi hybrydowymi środowiskami programistycznymi i uruchomieniowymi

udostępnianymi na najnowszych superkomputerach wspierając jednocześnie wysokopoziomowe i zorientowane dziedzinowo środowiska programistyczne jak opisane powyżej narzędzia CaKernel, QCG-OMPI oraz MUSCLE/2.



Rysunek 7. *Ogólna architektura systemu zarządzania zadaniami QCG w systemach komputerowych dużej mocy ze wsparciem dla hybrydowego środowiska programistycznego i uruchomieniowego QCG-OMPI oraz MUSCLE/2.*

W kontekście budowania interfejsów programistycznych wysokiego poziomu ostatnim bardzo ciekawym obszarem badań było opracowanie metod zarządzania zadaniami przetwarzającymi duże wolumeny danych [P10]. W odróżnieniu od typowych interfejsów programistycznych przyjęto założenie, że interfejsem do tworzenia i oceny eksperymentów obliczeniowych będą zrozumiałe dla człowieka frazy tekstu, a dziedzinową domeną będą duże wolumeny tekstów oraz problem wydajnej ekstrakcji słów kluczowych. Dla tak postawionego problemu opracowano model matematyczny oraz zaproponowano wielokryterialne metody ekstrakcji słów kluczowych jako zbiór zadań kaskadowych z ograniczeniami kolejnościowymi uruchamianymi na heterogenicznych węzłach dostępowych do danych. W pracy [P10] przedstawiono wyniki eksperymentów obliczeniowych oraz wielokryterialnej analizy wydajności obliczeń porównując różne środowiska programistyczne i uruchomieniowe, w tym sekwencyjne wykonanie obliczeń, równoległe wykonanie obliczeń oraz obliczenia z wykorzystaniem procedury uczenia maszynowego.

Omówienie wykorzystania wyników pracy naukowej oraz stworzonego oprogramowania

Wykorzystanie wyników pracy naukowej pozwoliło na podjęcie prac rozwojowych i wdrożeniowych oprogramowania dwóch rzeczywistych systemów informatycznych:

- systemu informatycznego symulatora GSSIM do badania wydajności algorytmów zarządzania zadaniami w hierarchicznych i heterogenicznych systemach komputerowych oraz modelowania i przeprowadzania eksperymentów obliczeniowych związanych z badaniem wydajności aplikacji oraz zaawansowanych eksperymentów obliczeniowych [GSSIM];
- systemu informatycznego QCG do zarządzania zadaniami w systemach komputerowych dużej mocy [QCG].

Pierwszy pakiet oprogramowania symulatora GSSIM początkowo wspierał tylko podstawowe mechanizmy niezbędne do modelowania i weryfikacji wydajności algorytmów zarządzania i szeregowania zadań w złożonych topologiach systemów komputerowych [P1]. W efekcie

prac badawczo-rozwojowych oprogramowanie GSSIM zostało rozbudowane o wiele różnych modułów dostarczających nowych funkcjonalności związanych z modelowaniem i testowaniem również wydajności różnych klas aplikacji. Oprogramowanie symulatora GSSIM pozwala na modelowanie i tworzeniem profili energetycznych dla aplikacji w odniesieniu do referencyjnych heterogenicznych architektur zasobów obliczeniowych połączonych w topologie oraz struktury hierarchiczne w dowolnej konfiguracji, np. uzależnione od typu i częstotliwości taktowania procesora oraz topologii superkomputera. Opracowywane profile aplikacji i heterogenicznych zasobów sprzętowych weryfikowane były na podstawie wielu testów wykonywanych w rzeczywistych hierarchicznych i heterogenicznych systemach [P2][P3][P5-9]. Warto zaznaczyć, że oprogramowanie symulatora GSSIM udostępnia użytkownikom implementacje wielu popularnych metod zarządzania zasobami i szeregowania zadań, które mogą być również w dowolny sposób rozbudowywane, testowane i weryfikowane z wykorzystaniem zarówno rzeczywistych danych pochodzących z logów systemów w różnych formatach, danych zbiorów testowych wykorzystywanych w testach wydajnościowych, jak i dowolnych danych generowanych syntetycznie. W efekcie stworzono referencyjną bazę oprogramowania oraz wyników dla różnych metod zarządzania zadaniami w systemach komputerowych dużej mocy z uwzględnieniem różnych architektur sprzętowo-programowych. Ponadto, eksperymenty symulacyjne przy użyciu oprogramowania GSSIM pozwalają aktualnie na przetestowanie metod zarządzania zasobami w skali, która nie jest możliwa przy wykorzystaniu środowisk testowych lub też symulacje architektur obliczeniowych superkomputerów nowych generacji na poziomie exa-skali, które będą wdrażane w najbliższej przyszłości.

Drugi pakiet oprogramowania opracowany i wykorzystywany w zaprezentowanych badaniach to zintegrowany zestaw usług i narzędzi QCG dla użytkowników i administratorów do zarządzania zasobami i zadaniami w środowisku obliczeniowym dużej mocy z pełnym wsparciem dla różnych scenariuszy aplikacji uruchamianych dużej skali, w tym aplikacji równoległych, aplikacji parametrycznych oraz złożonych aplikacji typu workflow. Oprogramowanie QCG umożliwia logiczne scalenie hierarchicznych i heterogenicznych zasobów obliczeniowych pochodzących z wielu klastrów obliczeniowych w jeden logicznie spójny, choć w praktycznych wdrożeniach często rozproszony system obliczeniowy pozwalający na uruchamianie różnego typu złożonych eksperymentów obliczeniowych o wymaganiach przekraczających możliwości pojedynczego klastra. Usługi QCG zintegrowane i przetestowane zostały z wiodącymi środowiskami programistycznymi i uruchomieniowymi dla aplikacji równoległych, w szczególności z omówionymi w cyklu prac środowiskami programistycznymi dla aplikacji stenciowych (CaKernel) i wielkoskalowych (MUSCLE/2). Ponadto, w ramach prac rozwoju oprogramowania oraz prac wdrożeniowych:

- dokonano znaczącego wkładu w opracowanie rozproszonej architektury rzeczywistego systemu komputerowego QCG składającego się z wielu zintegrowanych usług i narzędzi programistycznych;
- dokonano znaczącego wkładu w rozwój oprogramowania, a następnie pomyślny cykl wdrożeń systemu QCG do zarządzania zadaniami w rzeczywistych środowiskach produkcyjnych wszystkich Centrum Komputerów Dużej Mocy w kraju - infrastruktura nauki PL-Grid i systemy obliczeniowe dużej mocy o łącznej mocy > 8PFlops, odpowiednio w:
    - Poznaniu (PCSS);
    - Krakowie (Cyfronet);
    - Warszawie (ICM);
    - Gdańsku (TASK);
    - Wrocławiu (WCSS).

Przeprowadzono referencyjne testy i wdrożenia systemu komputerowego QCG do zarządzania zadaniami obliczeniowymi za granicą (infrastruktura PRACE i EGI klasy Tier 1-2 na poziomie powyżej 1PFlops), odpowiednio w:

- Garching (LRZ), Niemcy;
- Amsterdam (SARA), Holandia;
- Swindon (STFC), Wielka Brytania.

Kody źródłowe oprogramowania systemów komputerowych GSSIM oraz QCG są udostępnione na wolnej licencji, co umożliwia ich kontrolę i dalszy rozwój przez innych naukowców na świecie. Ponadto, symulator GSSIM oraz jego rozbudowana wersja DCWorms może zostać z powodzeniem wykorzystany do planowania i analizy nowych lub hipotetycznych konfiguracji przyszłych architektur sprzętowo-programowych w exa-skali niedostępnych aktualnie na rynku.

Podsumowanie

Badania przedstawione w ramach cyklu publikacji [P1]-[P10] wnoszą istotny wkład w rozwój dwóch dyscyplin informatyki systemów informatycznych dużej mocy oraz symulacji komputerowych demonstrując jednocześnie nowe interdyscyplinarne kierunki dalszych badań naukowych. Celem nadrzędnym wszystkich omówionych prac było zademonstrowanie kompleksowego i spójnego podejścia do problematyki metod zarządzania zadaniami w systemach komputerów dużej mocy, których architektury sprzętowo-programowe w ostatniej dekadzie ulegały i nadal ulegają znacznym zmianom. W wyniku przeprowadzonych badań wykazano słuszność innowacyjnego podejścia stworzenia dedykowanych narzędzi do modelowania i badania wydajności systemów komputerowych niezależnie od ich różnych konfiguracji, zaawansowanej hierarchicznej i heterogenicznej architektury zasobów, różnych klas aplikacji oraz różnych równoległych środowisk programistycznych i uruchomieniowych. Wszystkie eksperymenty obliczeniowe wykonane w rzeczywistych systemach dużej mocy oraz w środowisku symulatora GSSIM objęte zostały wnikliwą i przejrzystą analizą otrzymanych wyników, a w konsekwencji pozwoliły na usprawnienie i weryfikacje zarówno teoretycznych założeń w opracowanych modelach matematycznych, jak i wdrożenie nowych praktycznych rozwiązań w rzeczywistym systemie do zarządzania zadaniami QCG.

Podsumowanie wkładu naukowego zawartego w 10 publikacjach wchodzących w skład cyklu zaprezentowano poniżej:

- wypracowano metodologię tworzenia oraz wielokryterialnej oceny wydajności metod zarządzania zadaniami w hierarchicznych i heterogenicznych systemach komputerowych dużej mocy wraz z zaprojektowaniem, wdrożeniem i wykorzystywaniem zaawansowanego środowiska symulacyjnego GSSIM [P1];
- wypracowano nowe metody wielokryterialnej analizy problemu wraz z doborem parametrów kontrolnych dla konfiguracji różnych metod zarządzania zadaniami w hierarchicznej topologii systemu obliczeniowego łączącego poziom meta-systemu oraz poziom systemu kolejkowego [P4];
- zaproponowano efektywne metody do rozwiązywania praktycznych problemów zarządzania zadaniami z uwzględnieniem wielu ograniczeń, protokołów wymiany informacji o stanie zadań i zasobów w systemie komputerowym. W szczególności zaproponowano nowe metody modelowania i oceny wydajności systemu obliczeniowego oraz zaawansowanych aplikacji jako różnych typów zadań uruchamianych w hybrydowych i równoległych środowiskach programistycznych, w tym dla:
  - zadań stencilowych (równoległe środowisko programistyczne i uruchomieniowe CaKernel oraz nowe metody zarządzania aplikacjami stencilowymi na różnych zasobach heterogenicznych) [P2][P7][P9];

    - zadań wieloskalowych (równoległe środowisko programistyczne i uruchomieniowe MUSCLE/2) [P6];
    - zadań krytycznych [B17];
    - zadań przetwarzania dużych wolumenów danych [P10].
- wykonano szczegółowe analizy wyników eksperymentów w symulatorze GSSIM w celu optymalnego doboru parametrów kontrolnych dla metod zarządzania zadaniami dużej skali w rzeczywistych systemach na etapie prac przedwdrożeniowych [P8];
- wykazano eksperymentalnie, że dzięki zintegrowanej infrastrukturze obliczeniowej kilku Centrów Danych/HP na bazie opracowanego systemu QCG możliwa jest budowa środowiska programistycznego i uruchomieniowego dużej skali dla problemów wielkich wyzwań. Jest to jeden z istotnych kroków w kierunku planowania i budowy rozproszonych systemów obliczeniowych dużej mocy o łącznej wydajności poziomu exa-skali dla zaawansowanych symulacji i eksperymentów obliczeniowych [P6][P8];
- zaprezentowano nowe metody wydajnego zarządzania zadaniami w rozważanej klasie systemów z uwzględnieniem wielu ograniczeń i kryteriów oceny wydajności [P1][P9], w szczególności kryteriów związanych z:
    - poziomem wykorzystania zasobów obliczeniowych,
    - czasem wykonania oraz czasem oczekiwania zbioru zadań obliczeniowych uszeregowanych na hierarchicznych i heterogenicznych zasobach komputerowych,
    - zużycia energii dla wybranej klasy aplikacji i zadań obliczeniowych;
- opracowano nowe modele i metryki pomiarów wydajności aplikacji z uwzględnieniem zużycia energii dla przykładowej klasy aplikacji stencilowych w środowisku symulacyjnym i rzeczywistych systemach komputerów dużej mocy [P7].

Literatura

[Asa06] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley," Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep. UCB/EECS-2006-183, 2006

[Asa09] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, K. Yelick, : A view of the parallel computing landscape. Communications of the ACM 52(10), pp. 56–67, DOI: 10.1145/1562764.1562783, 2009

[Aug12] C. Augonnet, et al., StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators, In: Träff J.L., Benkner S., Dongarra J.J. (eds) Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol 7490. Springer, Berlin, Heidelberg, DOI: 10.1007/978-3-642-33518-1_40, 2012

[Bec07] P. Beckman, S. Nadella, N. Trebon, I. Beschastnikh, SPRUCE: A System for Supporting Urgent High-Performance Computing. In: Gaffney P.W., Pool J.C.T. (eds) Grid-Based Problem Solving Environments. IFIP The International Federation for Information Processing, vol 239. Springer, Boston, MA, DOI: 10.1007/978-0-387-73659-4_16, 2007

[Mah99] M. Maheswaran, et al., Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. J. Parallel Distrib. Comput. 59, 2 pp. 107-131. DOI=10.1006/jpdc.1999.1581, 1999

[Mar02] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller, and M. Upton. Hyper-threading technology architecture and microarchitecture," Intel Technology Journal, vol. 6, no. 1, pp. 4–15, 2002

[Sul04] A. Sulistio, C.S. Yeo and R. Buyya, A taxonomy of computer-based simulation and its mapping to parallel and distributed systems simulation tools, International Journal of Software: Practice and Experience 34 (2004), 653–673, DOI: 10.1002/spe.585, 2004

[Dee05] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, B. G. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems, Scientific Programming Journal, vol. 13, iss. 3, pp. 219-237, 2005

[Dee15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, Pegasus: a Workflow Management System for Science Automation, Future Generation Computer Systems, vol. 46, pp. 17-35, 2015.

[Cro12] D. Crooks et al., Multi-core job submission and grid resource scheduling for ATLAS AthenaMP, J. Phys.: Conf. Ser. 396 032115 doi:10.1088/1742-6596/396/3/032115, 2012

[For15] A. Forti, et al., Multicore job scheduling in the Worldwide LHC Computing Grid. Journal of Physics: Conference Series. 664. 062016. DOI: 10.1088/1742-6596/664/6/062016, 2015

[Kam99] S.-I. Kamata, R. O. Eason, and Y. Bandou, "A new algorithm for N-dimensional Hilbert scanning," IEEE Transactions on Image Processing, vol. 8, no. 7, pp. 964–973, 1999

[Gro99] W. Gropp, W. Lusk, and A. Skjellum (1999). Using MPI: portable parallel programming with the message-passing interface, volume 1. MIT Press

[Sul04] A. Sulistio, C.S. Yeo and R. Buyya, A taxonomy of computer-based simulation and its mapping to parallel and distributed systems simulation tools, International Journal of Software: Practice and Experience 34 (2004), 653–673

[Cha07] L. B. Chamberlain, D. Callahan, H. P. Zima, Parallel Programmability and the Chapel Language, International Journal of High Performance Computing Applications, 21(3): pp. 291-312, DOI: 10.1177/1094342007078442, 2007

[Cha08] B. Chapman, G. Jost, and R. Van Der Pas (2008). Using OpenMP: portable shared memory parallel programming, volume 10. MIT Press, 2008

[Bła07] J. Błażewicz, K.H. Ecker, E. Pesch, G. Schmidt and J. Węglarz, Handbook pf Scheduling: From Theory to Applications, Springer-Verlag, Berlin 2007

[Węg11] J. Węglarz, J. Józefowska, M. Mika and G. Waligóra, Project scheduling with finite or infinite number of activity processing modes: a survey, European Journal of Operational Research 2008, 117-205 (2011), DOI: 10.1016/j.ejor.2010.03.037, 2011

[Agu11] E. Agullo, C. Coti, T. Herault, J. Langou, S. Peyronnet, A. Rezmerita, F. Cappello, J. Dongarra, (2011). QCG-OMPI: MPI applications on grids. Future Generation Comp. Syst.. 27. 357-369, DOI: 10.1016/j.future.2010.11.015, 2011

[Car00] B. Carpenter, V. Getov, G. Judd, T. Skjellum and G. Fox. MPJ: MPI-like Message Passing for Java. Concurrency: Practice and Experience, Volume 12, Number 11. September 2000

[Bal11] F. Ballestin, R. Blanco, Theoretical and Practical Fundamentals for Multiobjective Optimisation in Resource-Cnstrained Prooject Scheduling Problems. Computers and Operations Research, 127 (2), pp. 297-316, DOI: 10.1016/j.cor.2010.02.004, 2011

[Tia12] K. Tian, Y. Jiang, X. Shen, W. Mao, Optimal Co-Scheduling to Minimize Makespan on Chip Multiprocessors. In: Cirne W., Desai N., Frachtenberg E., Schwiegelshohn U. (eds) Job Scheduling

Strategies for Parallel Processing. JSSPP 2012. Lecture Notes in Computer Science, vol 7698. Springer, Berlin, Heidelberg, DOI: 10.1007/978-3-642-35867-8_7, 2012

[Sel04] S. Sellappa and S. Chatterjee. Cache-efficient multigrid algorithms. International Journal of High Performance Computing Applications, 18(1):115–133, 2004

[Fri05] M. Frigo and V. Strumpen. Cache oblivious stencil computations. In Proceedings of the 19th annual international conference on Supercomputing, pages 361–366. ACM, 2005

[Dat09] K. Datta, S. Kamil, S. Williams, L. Oliker, J. Shalf, and K. Yelick. Optimization and performance modeling of stencil computations on modern microprocessors. SIAM review, 51(1):129–159, 2009

[Ngu10] A. Nguyen, N. Satish, J. Chhugani, C. Kim, and P. Dubey. 3.5-d blocking optimization for stencil computations on modern cpus and gpus. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–13. IEEE Computer Society, 2010.

[Tre11] J. Treibig, G. Wellein, and G. Hager. Efficient multicore-aware parallelization strategies for iterative stencil computations. Journal of Computational Science, 2(2):130–137, 2011

[Now13] M. Nowicki, P. Bała, PCJ-new approach for parallel computations in Java, In: P. Manninen, P. Oster (Eds.) Applied Parallel and Scientific Computing, LNCS 7782, Springer, Heidelberg (2013) pp. 115-125, 2013

[Tab13] L. G. Taboada, S. Ramos, R. Expósito, J. Touriño, R. Doallo, Java in the High Performance Computing arena: Research, practice and experience, Science of Computer Programming, Volume 78, Issue 5, DOI: 10.1016/j.scico.2011.06.002, 2013

[Tau02] D. S. Taubman and M. W. Marcellin, JPEG2000: Image Compression Fundamentals, Standard and Practice, Kluwer, Boston, 2002

[Imr03] Imreh, C. Computing 70: 277. https://doi.org/10.1007/s00607-003-0011-9, 2003

[Chen15] L Chen, D. Ye, G. Zhang, Approximating the Optimal Algorithm for Online Scheduling Problems via Dynamic Programming , Asia Pac. J. Oper. Res. 32, 1540011, https://doi.org/10.1142/S0217595915400114, 2015

[Rap15] Raphael Bleuse, Safia Kedad-Sidhoum, Florence Monna, Grégory Mounié, and Denis Trystram. 2015. Scheduling independent tasks on multi-cores with GPU accelerators. Concurr. Comput. : Pract. Exper. 27, 6 (April 2015), 1625-1638. DOI=http://dx.doi.org/10.1002/cpe.3359, 2015

[Ked15] S. Kedad-Sidhoum, F. Monna, D. Trystram, Scheduling Tasks with Precedence Constraints on Hybrid Multi-core Machines, IPDPSW 2015 – IEEE International Parallel and Distributed Processing Symposium Workshop, May 2015, Hyderabad, India. pp.27–33, 2015

[Mar17] L. Marchal, L. Canon, F. Vivien. Low-Cost Approximation Algorithms for Scheduling Independent Tasks on Hybrid Platforms. [Research Report] RR-9029, Inria - Research Centre Grenoble – Rhône-Alpes. 2017

[Per15] A. D. Pereira, L. Ramos, and L. F. Go ́es, “PSkel: a stencil programming framework for CPU-GPU systems, Concurrency and Computation: Practice and Experience, vol. 27, no. 17, pp. 4938–4953, 2015.

[Don15] J. Dongarra, M. A. Heroux, P. Luszczek "HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems," Technical Report, Electrical Engineering and Computer Sciente Department, Knoxville, Tennessee, UT-EECS-15-736, November, 2015.

[Pera15] S. Perarnau, et al. Distributed Monitoring and Management of Exascale Systems in the Argo Project. In: Bessani A., Bouchenak S. (eds) Distributed Applications and Interoperable Systems. DAIS 2015. Lecture Notes in Computer Science, vol 9038. Springer, 2015

[Phi11] S. C. Phillips, V. Engen, J. Papay, Snow White Clouds and the Seven Dwarfs, in: Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, CLOUDCOM '11, IEEE Computer Society, Washington, DC, USA, pp. 738–745, 2011

[Str12] J. A. Stratton, C. Rodrigrues, I.-J. Sung, N. Obeid, L. Chang, G. Liu, W.-M. W. Hwu, Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing, Tech. Rep. IMPACT-12-01, University of Illinois at Urbana-Champaign, 2012

[Evo15] Mc Evoy, G., Mury, A. R., and Schulze, B. An analysis of definition and placement of virtual machines for high performance applications on Clouds. Concurrency Computat.: Pract. Exper., 27, 1789–1814. doi: 10.1002/cpe.3346, 2015

[Orc10] D.A. Orchard, M. Bolingbroke and A. Mycroft, Ypnos: declarative, parallel structured grid programming, in: Proceedings of the 5th ACM SIGPLAN Workshop on Declarative Aspects of Multicore Programming, DAMP'10, 2010, pp. 15–24, 2010

[Una11] D. Unat, X. Cai and S.B. Baden, Mint: realizing CUDA performance in 3D stencil methods with annotated c, in: Proceedings of the International Conference on Supercomputing, ICS'11, pp. 214–224, 2011

[Pla09] J. Planas, R. Badia, E. Ayguad, J. Labarta, Hierarchical Task-Based Programming With StarSs. Int. J. High Perform. Comput. Appl. 23, 3 pp. 284-299, DOI:10.1177/1094342009106195, 2009

[Don11] J. Dongarra et al. The International Exascale Software Project Roadmap. IJHPCA, 25(1) pp. 3–60, DOI: 10.1177/1094342010391989, 2011

[Bik11] G. Bikshandi et al., Programming for Parallelism and Locality with Hierarchical Tiled Arrays, In Proc. 11th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming, ACM, pp. 48-57., 2011

[Kog13] P. M. Kogge, J. Shalf, Exascale computing trends: Adjusting to the "new normal"' for computer architecture, Computing in Science and Engineering, 15(6):16–26, 2013

[Lop16] R. V. Lopes, D. Menasce, A Taxonomy of Job Scheduling on Distributed Computing Systems, IEEE Transactions on Parallel and Distributed Systems (Volume: 27, Issue: 12, Dec. 1 2016), DOI: 10.1109/TPDS.2016.2537821, 2016

[Mem17] S. Memeti, et al., Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption. In Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC '17). ACM, New York, NY, USA, 1-6. DOI:10.1145/3110355.3110356, 2017

[Wae15] M. De Wael, S. Marr, B. De Fraine, T. Cutsem, W. De Meuter, Partitioned Global Address Space Languages. ACM Comput. Surv. 47, 4, Article 62 , 27 pp, DOI: 10.1145/2716320, 2015

[Fen12] W. Feng, Heshan Lin, T. Scogland, and J. Zhang. OpenCL and the 13 dwarfs: a work in progress. In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12). ACM, New York, NY, USA, pp. 291-294. DOI: 10.1145/2188286.2188341, 2012

[Xue14] W. Xue, C. Yang, H. Fu et al. Enabling and scaling a global shallow-water atmospheric model on Tianhe-2. in Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS '14), pp. 745–754, IEEE, Phoenix, Ariz, USA, 2014

[GSSIM] Oprogramowanie symulatora GSSIM/DCworms do modelowania procesów zarządzania zadaniami w hierarchicznych i heterogenicznych zasobach komputerowych: http://apps.man.poznan.pl/trac/gssim/wiki/DCWoRMS

[QCG] Oprogramowanie systemu zarządzania zadaniami i zasobami QCG dla systemów superkomputerowych: http://www.qoscosgrid.org/

[SWF] Parallel workload archive, http://www.cs.huji.ac.il/labs/parallel/workload/

[GWF] Grid workloads archive, http://gwa.ewi.tudelft.nl/

Krzysztof Kurowski