AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I ELEKTRONIKI
Katedra Informatyki

Autoreferat do rozprawy doktorskiej

Component-based Methodology for Programming and Running Scientific Applications on the Grid (Metodologia komponentowa do konstruowania i wykonywania aplikacji naukowych wykorzystujących zasoby gridowe)

mgr inż. Maciej Malawski

Promotor: prof. dr hab. inż. Jacek Kitowski

Kraków, 2009

1 Wprowadzenie

We współczesnych badaniach coraz większą rolę odgrywają *nauki obliczeniowe*, wykorzystujące komputery do modelowania, symulacji oraz analizy danych w dużej skali (*system-level science*) [3]. Przeanalizowane aplikacje naukowe wymagają dużych nakładów obliczeniowych i dużych ilości danych, realizują dynamiczne scenariusze użycia (eksperymenty) oraz wykorzystują różne stopnie rozproszenia i rodzaje kompozycji. Dodatkowo, często wymagają współpracy zespołów na wielu etapach konstruowania i uruchamiania, są tworzone w wielu językach programowania, a także w wielu przypadkach są budowane od podstaw przez naukowców programujących swoje modele i algorytmy.

Infrastrukturą, mającą na celu stanowić platformę do rozwoju e-nauki, są *systemy gridowe* [2]. Dostarczają one zasobów obliczeniowych, a także umożliwiają skoordynowane współdzielenie zasobów pozostających poza scentralizowaną kontrolą. Wśród istniejących infrastruktur gridowych na uwagę zasługują takie projekty jak EGEE, DEISA, Grid'5000, czy TeraGrid. Charakterystyczne własności systemów gridowych to rozproszenie, heterogeniczność i dynamika współdzielonych zasobów, nad którymi użytkownik nie posiada pełnej kontroli. Dodatkowe trudności wynikają z tego, że nie istnieje wspólny rodzaj oprogramowania warstwy pośredniej *middleware*, zaś infrastruktury są budowane w oparciu o różne koncepcje i oferowane modele programowania.

Z wymienionych powodów odpowiedź na pytanie *Co należy zrobić, aby programować i uruchamiać aplikacje naukowe na zasobach gridowych?* stanowi ważne i interesujące zagadnienie naukowe, pomimo wielu prac badawczych prowadzonych w tym zakresie. Odpowiedzią na tak postawione pytanie powinna być *metodologia*, będąca zbiorem metod i narzędzi, docelowo zintegrowanych w *środowisko* wspomagające proces budowania i wykonywania aplikacji naukowych na zasobach gridowych. W ramach prac zdefiniowano wymagania stawiane takiemu środowisku, w tym ułatwianie programowania na wysokim poziomie, ułatwienie rozmieszczania aplikacji na współdzielonych zasobach, dostosowanie do różnej skali zasobów, komunikacja dostosowana do różnych stopni rozproszenia, wspieranie wielu języków programowania, dostosowanie do środowiska gridowego podatnego na zmiany, a także współoperatywność.

2 Teza pracy

Opierając się na definicji problemu badawczego, autor postawił następującą tezę pracy:

> Użycie modelu komponentowego, wzbogaconego narzędziami programowania wyższego poziomu i połączonego z właściwą warstwą wirtualizacji, stanowi metodologię efektywnie wspomagającą programowanie i uruchamianie złożonych aplikacji naukowych na zasobach gridowych.

W celu weryfikacji tak postawionej tezy zdefiniowano podstawowy cel pracy, jakim było opracowanie metod i narzędzi do programowania i wykonywania aplikacji naukowych na gridzie. Szczegółowe prace badawcze obejmowały analizę modeli programowania dla aplikacji gridowych, określenie wymaganych cech środowiska programowania, implementację prototypu i studium wykonalności oraz weryfikację metod i narzędzi przy pomocy typowych aplikacji.

3 Aplikacje naukowe na gridzie – analiza rozwiązań

Sformułowanie propozycji metod i narzędzi stało się możliwe dzięki uprzedniej analizie istniejących podejść i rozwiązań. Przeanalizowano modele programowania, oprogramowanie umożliwiające dostęp do zasobów obliczeniowych, sposoby instalacji aplikacji (deployment), rozwiązania dotyczące współoperatywności oraz zarządzania aplikacją i jej adaptowalnością w trakcie działania.

Na podstawie analizy istniejących rozwiązań otrzymano następujące wnioski:

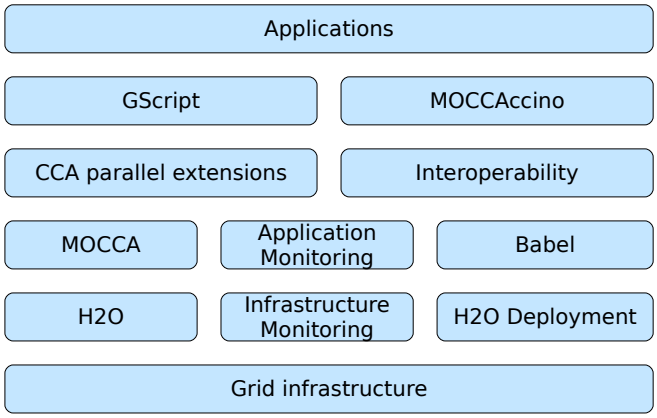
- Wiele z postawionych wymagań może zostać spełnionych jeżeli jako bazowy model programowania zostanie wybrany *model komponentowy*.
- Istniejące środowiska komponentowe takie jak XCAT, CCAFFEINE, GridCCM, ASSIST, HOC-SA, ProActive a także *Web services* spełniają postawione wymagania tylko częściowo.

4 Metodologia komponentowa

Na podstawie przeprowadzonej analizy zaproponowano podejście opierające się na dwóch elementach:

- Przyjęcie modelu komponentowego jako bazowego modelu programowania.
- Użycie lekkiej platformy umożliwiającej wirtualizację zasobów.

W celu wykazania przydatności zaproponowanego rozwiazania podjęto decyzję o skoncentrowaniu się na następujących konkretnych technologiach: wybór **CCA** [1] jako modelu komponentowego oraz użycie platformy **H2O** [4] jako technologii zapewniającej wirtualizację zasobów. Dzięki tej decyzji, część z wymagań może być spełniona automatycznie, natomiast pozostałe elementy stały się tematem prac badawczych.



Rysunek 1: Warstwowa architektura zaproponowanego środowiska

Programowanie na wysokim poziomie abstrakcji Jako jedno z rozwiązań zaproponowano rozszerzenie mechanizmów kompozycji aplikacji w CCA o *programowanie skryptowe wysokiego poziomu*, zrealizowane jako element środowiska GridSpace i przydatne do dynamicznie tworzonych eksperymentów [7]. W zaproponowanym podejściu tworzenie komponentów oraz przepływ sterowania w aplikacji komponenotwej mogą być zapisane przy pomocy skryptu (GScript), zaś programista nie musi podawać szczegółów technicznych związanych z infrastrukturą – niskopoziomowe decyzje są podejmowane przez środowisko. Jako sposób konstruowania aplikacji alternatywny względem podejścia skryptowego zaproponowano użycie deskryptorów ADL (Architecure Description Language), dostosowanych do aplikacji o stabilnej strukturze. Podejście to polega na stworzeniu opisu aplikacji w postaci hierarchicznego dokumentu opartego na XML. Deskryptor ten jest następnie przetwarzany przez system menedżera aplikacji (MOCCAccino), który rozmieszcza komponenty na zasobach i uruchamia aplikację, a także może reagować na zmiany środowiska.

Ułatwienie rozmieszczania na współdzielonych zasobach Jest ono możliwe dzięki wykorzystaniu mechanizmów kernela H2O i dynamicznego ładowania aplikacji. Dodatkowo, zaproponowano metody tworzenia dynamicznej puli takich kontenerów na infrastrukturze gridowej [6], przykrywając w ten sposób gridową warstwę *middleware*.

Dostosowanie do różnej skali zasobów jest możliwe dzięki połączeniu cech lekkiego kernela H2O i możliwości rekonfiguracji aplikacji oferowanych przez model CCA. Dzięki temu aplikacje komponentowe można uruchamiać zarówno na pojedynczych maszynach, jak i na rozległych zasobach gridowych.

Dostosowanie komunikacji do różnych stopni rozproszenia jest możliwe dzięki wykorzystaniu biblioteki RMIX dostarczanej przez H2O. W celu ułatwienia komunikacji pomiędzy równolegle działającymi komponentami zaproponowano mechanizmy łączenia kolekcji komponentów.

Wspieranie aplikacji wielojęzycznych jest możliwe dzięki użyciu narzędzia Babel dostarczonego z CCA. Przeprowadzono integrację systemów Babel i RMIX uzyskując środowisko łączące wiele

języków programowania i wiele protokołów komunikacyjnych.

Dostosowanie do środowiska gridowego podatnego na zmiany Część z aspektów rekonfigurowalności i adaptowalności jest zapewniona dzięki połączeniu cech CCA i H2O. Elementy adaptowalności są zawarte w zaproponowanym systemie MOCCAccino, jak również poprzez podsystemy monitorujące i optymalizujące w środowisku GridSpace.

Współoperatywność Zaproponowano podejście umożliwiające współdziałanie i integrację aplikacji komponentowych opartych o modele CCA i GCM [5], a także opisano możliwości wykorzystania technologii Web services.

Struktura zaproponowanej koncepcji środowiska jest przedstawiona na Rys. 1. Powyżej niskopoziomowych zasobów gridowych znajduje się platforma H2O wraz z mechanizmami dynamicznego rozmieszczania kontenerów oraz monitorowaniem zasobów. Wyżej występuje MOCCA [8] będąca implementacją modelu CCA przy użyciu H2O, z takimi rozszerzeniami jak monitorowanie aplikacji i wspieranie komponentów tworzonych z użyciem systemu Babel. Kolejną warstwą są rozszerzenia wspomagające równoległe łączenie komponentów i współoperatywność. Środowiska programowania wysokiego poziomu GridSpace i MOCCAccino znajdują się bezpośrednio pod warstwą aplikacji.

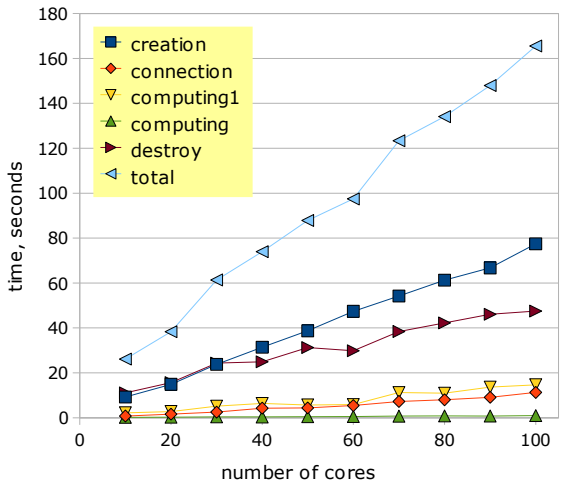
5 Aplikacje i testy

W celu weryfikacji zaproponowanych metod oraz narzędzi przeprowadzono szereg testów i eksperymentów. Użyto w nich przykładowych aplikacji naukowych zaprogramowanych zgodnie z modelem komponentowym, a także zestawu syntetycznych aplikacji testowych mających na celu zademonstrować wybrane cechy stworzonego środowiska.

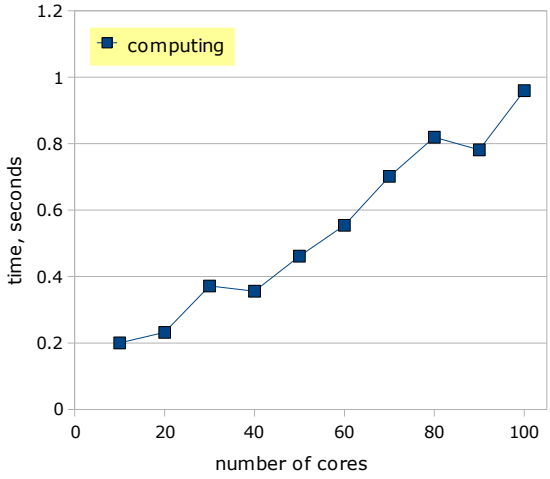
Do sprawdzenia poprawności działania środowiska MOCCA wykorzystano aplikację AFC stworzoną wcześniej dla środowiska XCAT. Wynik testu pokazał, że przeniesienie aplikacji z XCAT do MOCCA jest łatwe i wymaga jedynie niewielkich zmian wynikających głównie z innego sposobu uruchamiania aplikacji. Aplikacją obliczeniową, napisaną od podstaw, była symulacja powstawania klastrów z atomów złota metodą *simulated annealing*. Została ona z powodzeniem uruchomiona zarówno na lokalnych klastrach obliczeniowych jak i na infrastrukturze gridowej CrossGrid i Grid'5000. Kolejnymi aplikacjami demonstrującymi zalety podejścia skryptowego były eksperymenty *data mining* z użyciem biblioteki Weka, której moduły zostały opakowane jako komponenty MOCCA. Jako przykład aplikacji o wysokim stopniu powiązania (*tightly-coupled*) posłużyły obliczenia z użyciem automatów komórkowych zrównoleglone metodą dekompozycji domenowej. Testy wykazały skalowalność aplikacji uruchomionej na klastrze.

W celu zbadania zachowania środowiska MOCCA przy dużej liczbie węzłów obliczeniowych, przy pomocy syntetycznej aplikacji przetestowano proces rozmieszczania komponentów i wykonywania aplikacji na infrastrukturze Grid'5000. W skład infrastruktury wchodziło 13 klastrów obliczeniowych w 6 ośrodkach we Francji.

Celem pierwszego testu było sprawdzenie, jak zachowuje się aplikacja w przypadku *sekwencyjnego* wywoływania operacji na komponentach (wersja 1). Otrzymano zależność zbliżoną do liniowej, co potwierdza że obsługa wielkiej liczby portów i połączeń przez komponenty nie powoduje spadku wydajności. Celem drugiego testu było sprawdzenie, jak zachowa się aplikacja gdy wywoływanie metod na kolekcji komponentów będzie odbywało się współbieżnie z użyciem puli wątków (wersja 2 i 3 aplikacji). Rys. 2(a) i 2(b) przedstawiają pomiary czasów poszczególnych etapów aplikacji. Po-



(a) Pomiary czasów trwania poszczególnych etapów



(b) Średni czas wywołania metod na kolekcji komponentów (powiększony fragment rysunku z lewej)

Rysunek 2: Pomiary czasu trwania działania aplikacji (wersja 3) na 100 rdzeniach procesorów w 4 klastrach

równując zmierzone czasy z średnim opóźnieniem (*latency*) sieci między klastrami, stwierdzono że narzuty wprowadzane przez system komponentowy są akceptowalne.

Dodatkowo, przeprowadzono eksperymenty w których udało się uruchomić aplikację testową na 600 i 800 węzłach obliczeniowych (rdzeniach) w 8 klastrach.

Główny wniosek wynikający z eksperymentów przeprowadzonych na infrastrukturze Grid'5000 to potwierdzenie poprawnego działania środowiska MOCCA w wypadku wykorzystania dużej liczby zasobów i komponentów, co było istotnym wymaganiem zdefiniowanym w pracy.

Oprócz zamieszczonych w tym miejscu wyników, w pracy opisano również testy współoperatywności między GCM i CCA na przykładzie środowisk MOCCA i ProActive, badanie przydatności optymalizacji w środowisku GridSpace, a także przetestowano wydajność warstwy komunikacyjnej w środowisku MOCCA. W celu przetestowania zaproponowanej metody tworzenia wirtualnej puli kerneli H2O wykonano eksperymenty z aplikacją *simulated annealing* na infrastrukturze CrossGridu oraz Grid'5000.

Szeroki zakres uruchomionych aplikacji, przedstawionych w pracy, świadczy o przydatności zaproponowanego podejścia komponentowego. Dodatkowo, testy pokazujące działanie elementów środowiska na różnorodnych infrastrukturach, demonstrują że prototypowe narzędzia efektywnie wspomagają programowanie i uruchamianie aplikacji na zasobach gridowych.

6 Podsumowanie

Celem pracy było zaproponowanie metodologii programowania i uruchamiania aplikacji naukowych na zasobach gridowych. W pracy autor pokazał w jaki sposób tę metodologię można zrealizować łącząc zalety modelu komponentowego CCA i platformy H2O, stanowiące podstawę środowiska do programowania i uruchamiania aplikacji. Do rozszerzeń przedstawionych przez autora należy zaliczyć podejścia do komponowania i uruchamiania aplikacji na wysokim poziomie abstrakcji przy

użyciu języka skryptowego lub deskryptorów aplikacji, łączenie wielu języków programowania z użyciem systemu Babel, współdziałanie modeli komponentowych CCA i GCM, a także metody uruchamiania aplikacji na infrastrukturach gridowych. Środowisko komponentowe MOCCA, stworzone przez autora, stanowi bazę dla tych rozwiązań, a także dla aplikacji użytych do zweryfikowania zaproponowanej metodologii. Wyniki opisane w pracy potwierdziły prawdziwość postawionej przez autora tezy.

Głównym wkładem pracy badawczej autora jest zaproponowany zestaw metod i narzędzi opartych o model komponentowy, z włączeniem następujących elementów:

- Autor opracował *koncepcje* wysokopoziomowego podejścia skryptowego oraz opartego na ADL wspierającego konstruowanie aplikacji komponentowych, środowiska komponentowego MOCCA łączącego cechy CCA i H2O, rozwiązania współoperatywności pomiędzy CCA i GCM, współdziałania wielu języków i wielu protokołów komunikacyjnych przy użyciu Babel i RMIX, a także metodę tworzenia puli kontenerów komponentowych w celu rozmieszczania komponentów na infrastrukturach gridowych.

- W celu weryfikacji zaproponowanych koncepcji autor zaproponował i *zaprojektował* następujące narzędzia i rozwiązania: projekt wspierania komponentów CCA w środowisku skryptowym GridSpace, architekturę menedżera MOCCAccino, projekt środowiska komponentowego MOCCA, projekt rozwiązania łączącego CCA i GCM, projekt integracji systemów RMIX i Babel.

- Następujące aspekty środowiska były przedmiotem prototypowej *implementacji* zrealizowanej przez autora: środowisko komponentowe MOCCA, adaptery dla technologii MOCCA w środowisku GridSpace, implementacja rozwiązania łączącego CCA i GCM przy użyciu MOCCA i ProActive oraz narzędzia wspomagające rozmieszczanie komponentów na infrastrukturze gridowej.

- Aby zweryfikować zrealizowane prototypowe rozwiązania, autor przeprowadził *eksperymenty* i testy z użyciem przykładowych aplikacji naukowych.

Prace badawcze stanowiły część badań w ramach zespołu Katedry Informatyki i ACK-CYFRONET AGH, przy szerokiej współpracy zagranicznej z takimi ośrodkami jak Emory University w Atlancie i INRIA Sophia-Antipolis, a także stanowiły wkład do projektów europejskich CoreGRID i ViroLab.

W efekcie pracy określono następujące interesujące możliwości dalszego rozwoju badań: rozwój algorytmów wspierających zaproponowane metody, rozszerzenie o koncepcje *Semantic Web*, ściślejsza integracja zaproponowanego środowiska, opracowanie modelu formalnego, poszerzenie wspierania wielu języków programowania, integracja systemu bezpieczeństwa.

Literatura

[1] R. Armstrong, G. Kumfert, L. C. McInnes, S. Parker, B. Allan, M. Sottile, T. Epperly, and T. Dahlgren. The CCA component model for high-performance scientific computing. *Concurrency and Computation : Practice and Experience*, 18(2):215–229, 2006.

[2] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global*

Grid Forum, 2002. `http://www.globus.org/alliance/publications/papers/ogsa.pdf`.

[3] I. Foster and K. Kesselman. Scaling system-level science: Scientific exploration and IT implications. *Computer*, 39(11):31–39, 2006.

[4] D. Kurzyniec et al. Towards Self-Organizing Distributed Computing Frameworks: The H2O Approach. *Parallel Processing Lett.*, 13(2):273–290, 2003.

[5] M. Malawski, M. Bubak, F. Baude, D. Caromel, L. Henrio, and M. Morel. Interoperability of grid component models: GCM and CCA case study. In *Towards Next Generation Grids, proceedings of the CoreGRID Symposium in conjunction with Euro-Par 2007*, CoreGRID series, pages 95–106. Springer, August 2007.

[6] M. Malawski, M. Bubak, M. Placek, D. Kurzyniec, and V. Sunderam. Experiments with distributed component computing across grid boundaries. In *Proceedings of the HPC-GECO/CompFrame workshop in conjunction with HPDC 2006*, Paris, France, 2006.

[7] M. Malawski, T. Gubala, M. Kasztelnik, T. Bartyński, M. Bubak, F. Baude, and L. Henrio. High-level scripting approach for building component-based applications on the grid. In *CoreGRID Workshop on Grid Programming Model Grid and P2P Systems Architecture Grid Systems, Tools and Environments*, Heraklion, Crete, June 2007. Springer.

[8] M. Malawski, D. Kurzyniec, and V. Sunderam. MOCCA – towards a distributed CCA framework for metacomputing. In *Proceedings of the 10th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS2005) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2005)*. IEEE Computer Society, 2005.