Prof. dr hab. inż. **Kazimierz Subieta**
Polsko-Japońska Wyższa Szkoła Technik Komputerowych
Wydział Informatyki
ul. Koszykowa 86, 02-008 Warszawa

Warszawa, 29 stycznia 2009 r.

Recenzja

rozprawy doktorskiej mgr inż. **Marka Grochowskiego**

"Object- and Agent-Oriented Strategies for High Performance Computing in a Computer Network"

(Obiektowe i agentowe strategie przetwarzania wielkich zadań obliczeniowych w sieciach komputerowych)

1. Dane formalne rozprawy

- **Promotor:** Prof. dr hab. inż. **Robert Schaefer**
- **Instytucja przeprowadzająca przewód doktorski**: Rada Wydziału Informatyki Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych
- **Rozmiar treści rozprawy**: 132 strony, w tym
  - 86 stron zasadniczej treści,
  - 14 stron edycyjnych (nagłówki, spisy, listy, streszczenia, podziękowania)
  - 10 stron cytowanej literatury (138 pozycji, w tym 13 pozycji, gdzie autor rozprawy jest autorem lub współautorem, 12 pozycji w języku angielskim, 1 w języku polskim)
  - 22 strony załączników
- Rozprawa składa się z 8-miu rozdziałów merytorycznych, podsumowania, 2-ch załączników i bibliografii.
- Rozprawa jest napisana w języku angielskim, z jednostronicowym streszczeniem w języku polskim.

2. Potwierdzenie naukowego znaczenia badań Autora rozprawy na forum międzynarodowym

Oprócz samej rozprawy, jej tematyka jest przedmiotem 12 artykułów o znaczeniu międzynarodowych, większość z listy MNiSzW (zwanej listą filadelfijską), których współautorem był mgr Marek Grochowski. Popularny portal bibliograficzny DBLP (http://www.informatik.uni-trier.de/~ley/db/) zawierający informację o publikacjach z dziedziny informatyki o znaczeniu międzynarodowym zawiera 13 pozycji gdzie Marek Grochowski był współautorem, z czego 7 pozycji dotyczy tematyki rozprawy. (Pozostałe pozycje dotyczą być może innej osoby o takim samym imieniu i nazwisku – nie udało mi się tego ustalić.) Ten wynik nie jest typowy dla doktorantów i dowodzi zarówno wysokiej rangi przeprowadzonych badań, jak i predyspozycji doktoranta do aktywnego udziału w nauce światowej. Powinien być bardzo wysoko oceniony.

3. Poziom techniczny i edytorski rozprawy

Poziom techniczny i edytorski rozprawy nie budzi zastrzeżeń. Poziom języka angielskiego rozprawy uważam za bardzo dobry - poza kilkoma drobnymi przypadkami nie zauważyłem istotnych uchybień. Układ treści rozprawy nie budzi zastrzeżeń. Definicje pojęć, terminów i oznaczeń są kompletne i poprzedzają ich użycie. W przypadku, gdyby rozprawa była zakwalifikowana do druku, uważałbym za konieczne dopracowanie rozprawy tak, aby stała się bardziej czytelna dla powszechnego użytkownika, poprzez dobrze dobrane przykłady i większą przyjazność do czytelnika w przedstawianiu motywacji dla niektórych definicji, wywodów i metod.

4. Końcowa konkluzja dotycząca rozprawy

Moja ocena rozprawy mgr Marka Grochowskiego jest pozytywna.

Uważam, że rozprawa ta spełnia warunki ustawy o stopniu naukowym doktora i wnoszę o dopuszczenie jej do dalszych kroków w przewodzie doktorskim i do publicznej obrony.

5. Charakterystyka treści rozprawy i przeprowadzonych badań

Przedmiotem rozprawy jest rozproszona infrastruktura informatyczna niezbędna dla realizacji wielkich zadań obliczeniowych o jednorodnym charakterze obliczeń. Obliczenia tego typu, w szczególności metoda elementów skończonych czy też obliczenia związane z optymalizacją przy użyciu algorytmów genetycznych, mają ważne zastosowania w wielu działach nauki i techniki. Celem zasadniczym tej infrastruktury jest poprawienie szybkości obliczeń. Same metody obliczeniowe są znane i nie są rezultatem przeprowadzonych badań, jakkolwiek posłużyły do eksperymentów. Dla osób takich jak recenzent tej rozprawy, zajmujących się głównymi nurtami informatyki narzędziowej, takimi jak inżynieria oprogramowania, języki programowania, bazy danych, technologie Internetu, portale biznesowe itp. zastosowania tego typu wydają się rzadkie, nawet marginalne. Niemniej nawet rzadkie zastosowania mogą być bardzo ważne, gdyż na podstawie takich obliczeń można podejmować decyzje techniczne o skali dziesiątków lub setek milionów złotych. Uznaję za dość wiarygodną tezę, że poprawa szybkości obliczeń przy pomocy nowej infrastruktury informatycznej jest na tyle istotna, że uzasadniało podjęcie badań w ramach przedmiotowej pracy doktorskiej.

To co jest niezwykłe w tego rodzaju zastosowaniach jest bardzo duża regularność obliczeń umożliwiająca dekompozycję dużych zadań obliczeniowych na setki lub tysiące małych procesów, które mogą wykonywać się równolegle na wielu fizycznych komputerach. Typowe zadania o charakterze biznesowym są pod tym względem o wiele bardziej wymagające, zaś masowe zrównoleglenie procesów nadal jest bardzo poważnym problemem. Przykładem jest japoński projekt komputerów tzw. 5-tej generacji (1982-1992), który dotyczył masowej równoległości przetwarzania w bazach danych i zakończył się rezultatami zbliżonymi do zera, przy nakładach na projekt rzędu kilku miliardów dolarów. Obecnie z podobnym problemem borykają się technologie super-komputerów (które okazały się nieefektywne dla masowych zastosowań w biznesie) oraz technologie określane jako klastry lub grid. We wszystkich tych przypadkach masowa równoległość jest fizycznie możliwa, ale technicznie trudna do wykorzystania z kilku powodów, m.in. z takiego, że obecne technologie programistyczne oparte głównie na jednowątkowych algorytmach, rzadko dają się w spójny sposób zrównoleglić obliczeniowo. Niemniej fizyczne zrównoleglenie tego typu zadań występuje w specyficznych środowiskach, takich jak sieci P2P lub technologie procesów pracy

(workflow). Niektóre duże zadania obliczeniowe również mogą napotkać tego rodzaju problem z masową równoległością.

Autor rozprawy omawia trzy główne podejścia do masowej równoległości przetwarzania w dużych zadaniach obliczeniowych. Jest to podejście niskopoziomowe nastawione na zapis obliczeń w językach takich jak C lub Fortran, wspomaganych przez wyspecjalizowane biblioteki i protokoły zapewniające równoległe przetwarzania. Generalna konkluzja Autora zmierza w kierunku stwierdzenia, że ten rodzaj programowania prowadzi do efektywnego rozwiązania problemu, ale jest dość uciążliwy (kosztowny) przy programowaniu i pielęgnacji aplikacji. Drugim podejściem jest podejście obiektowe bazujące na narzędziach obiektowych takich jak język Java oraz oprogramowanie pośredniczące (*middleware*) bazujące na rozwiązaniach znanych ze standardu CORBA lub na technologii RMI (Java). Zaletą tego podejścia jest uwolnienie się (przynajmniej częściowo) od problemów związanych z rozproszeniem i heterogenicznością platform, na których dokonuje się obliczeń. Dzięki własnościom obiektowym, takim jak obiekty, klasy, dziedziczenie, hermetyzacja, polimorfizm, łatwiejsze jest abstrakcyjne sformułowanie problemu, ponowne użycie kodu oraz modelowanie pojęciowe podczas programowania. Te zalety stanowią przewagę nad podejściem niskopoziomowym.

Trzecim podejściem do wielkich zadań obliczeniowych, najbardziej perspektywicznym z punktu widzenia Autora rozprawy, jest podejście multi-agentowe. W podejściu tym obliczenia są dekomponowane pomiędzy agentów, posiadających pewną dozę autonomii oraz zdolność do dzielenia się na mniejszych pod-agentów oraz do przemieszczenia się pomiędzy platformami obliczeniowymi. Podejście agentowe jest głównym motywem tej rozprawy i paradygmatem, w którym jej Autor uzyskał najbardziej znaczące wyniki. Wyniki te są zarówno rezultatem doświadczeń Autora przy implementacji tego rodzaju aplikacji za granicą (Szwajcaria) oraz budowy własnej platformy agendowej Octopus (zbudowanej w większym zespole). Rezultaty te dotyczą szeregu aspektów takiej platformy, takich jak jej formalny opis, podział agentów, protokoły ich migracji, opracowanie architektury systemu, itp.

Głównym rezultatem przedstawionym w dysertacji jest protokół migracji agentów, który jest analogią do dyfuzji molekularnej zachodzącej w kryształach. W dużym skrócie, protokół ten przypisuje pewne wagi (zapotrzebowanie na obliczenia) poszczególnym agentom. Agent przemieszcza się na taką platformę, gdzie istnieją rezerwy obliczeniowe biorąc pod uwagę

sumę wag znajdujących się tam agentów. Metoda została zapisana przy pomocy dość złożonego formalizmu, jakkolwiek nie ma dowodu, że istotnie ta metoda prowadzi do optymalizacji przydziału zadań na poszczególne jednostki wykonawcze i w efekcie, minimalizację czasu obliczeń. Są przedstawione jednak wyniki eksperymentów pokazujących niektóre zalety tej metody nad innymi metodami.

Rozprawa prezentuje wyniki eksperymentów dla różnych zadań obliczeniowych charakteryzujących się regularną i nieregularną równoległością. Przedstawiono szczegółowe opisy tych eksperymentów oraz wnioski, które na ich podstawie można było wyciągnąć. Dodatki zawierają krótkie opisy analizowanych grup obliczeń oraz opisy konkretnych metod numerycznych. Ponieważ jest to tematyka całkowicie mi obca, nie byłem w stanie zweryfikować zawartości tych załączników.

W podsumowaniu i konkluzjach Autor rozprawy podkreśla kilka osiągniętych wyników naukowych. Wśród nich znajduje się synteza algorytmu on line do planowania zadań w systemach rozproszonych, opracowanie typów obiektowych i komponentów do generowania siatki (mesh) obliczeniowej, zaprojektowanie i formalny opis platformy multi-agentowej, architektura określona jako Smart Solid Agent, algorytm do podziału agentów, algorytm migracji agentów oparty na analogii z dyfuzją molekularną w kryształach, i inne. Bez wątpienia, rezultaty te są oryginalne i posiadające odpowiednią rangę z punktu widzenia twórczości naukowej.

6. Uwagi do rozprawy i jej kontekstu

6.1. Biorąc pod uwagę to, że obecnie również pracuję nad projektem mającym na celu masową równoległość (procesów pracy), moje wątpliwości dotyczą uniwersalności proponowanej infrastruktury obliczeniowej. Chodzi o generalną charakterystykę zadań, które mogą być realizowane na tej infrastrukturze oraz zadań, które z pewnością dla tej infrastruktury się nie nadają. W kilku miejscach Autor rozprawy sugeruje uniwersalność tej infrastruktury dla wielkich zadań obliczeniowych, co prawdopodobnie nie jest prawdą (biorąc pod uwagę moje doświadczenia z równoległością). Wydaje się, że przydałby się na wstępie jakiś akapit na ten temat. Niemniej uznaję, że podane w rozprawie przykłady obliczeń w pełni uzasadniają stworzenie infrastruktury proponowanej przez Autora.

6.2. Kolejna wątpliwość dotyczy również moich doświadczeń z równoległością. Jednym z czynników stanowiących bardzo poważną przeszkodę w równoległości jest konieczność skumulowania lokalnych obliczeń w jedną całość przez jakiś scentralizowany podmiot. W przedmiotowej rozprawie agenci obliczają lokalnie, ale nie doczekałem się dostatecznie wyraźnego miejsca mówiącego o tym, w jaki sposób wyniki lokalnych obliczeń są konsumowane przez wspomniany podmiot. To staje się szczególnie ważne, gdy założymy, że dany agent może podzielić się na kilka agentów. W tym przypadku musi zachodzić semantyczna równoważność: wynik produkowany przez agenta przed podziałem musi być taki sam, jak sumaryczny wynik po podziale. Ponadto powstają problemy synchronizacyjne, czyli ustalenie warunków w których podmiot nadrzędny w stosunku do agentów może kontynuować swoją pracę. Niestety te sprawy pozostały w rozprawie niejasne.

6.3. Możliwość dowolnego przemieszczania się agentów i ich podziału jest podporządkowana wyłącznie optymalizacji obciążeń serwerów oraz minimalizacji globalnego czasu obliczeń. W systemach, którymi się zajmuję, są inne kryteria, takie jak dostępność zasobów lub aplikacji na danym serwerze, kwestie bezpieczeństwa, kwestie minimalizacji obciążenia sieci komputerowej, niezawodności, itd. Te nie zawsze łatwe problemy nie są dyskutowane w rozprawie, prawdopodobnie z powodu zawężenia jej przedmiotu do bardzo wąskiej klasy zadań.

6.4. Moim zdaniem, wstęp do pracy (Motivation) jest za krótki. Powinien szerzej przedstawić kontekst pracy, jej motywację, podstawowe pojęcia i główne rezultaty. Należy pamiętać, że często wstęp jest reklamą dla pracy. O ile nie będzie dostatecznie informacyjny, potencjalny czytelnik nie zechce dalej pracy czytać.

6.5. W wielu miejscach czytelnik może mieć problemy z rozszyfrowanie zamieszczonych formuł matematycznych. Np. na str.10 znajduje się sformułowanie „scheduling problem" w postaci pewnej notacji, która niestety nigdzie nie została objaśniona. Być może należy ona do folkloru dziedziny obliczeń numerycznych, ale pracę mogą czytać ludzie nie znający tego folkloru.

6.6. Na stronie 11 Autor rozprawy pisze w superlatywach na temat systemów agendowych, formułując tezę, że są one szeroko stosowane do ważnych rozproszonych zastosowań. Sądzę, że na te fanfary jest jeszcze za wcześnie. Po różnych pochopnych zapowiedziach i różnych doświadczeniach świat biznesowy bardzo krytycznie odnosi się do tego rodzaju technologii.

Jak dotąd, nie potwierdziła ona poprzez „killer application", że ma jakościową lub ilościową przewagę nad swoimi konkurentami (takimi chociażby jak paradygmat Serwis Oriented Architecture lub architektura multi-client/multi-server). W moim środowisku (odpowiedzialnych zastosowań biznesowych) głoszony jest wręcz pogląd, że technologie agentowe nic nie dają, bazują na powierzchownych antropomorfizmach rodem z dziennikarskiego magla i tak naprawdę są tylko jeszcze jednym wytrychem, przy pomocy którego pracownicy naukowi próbują dostać się do komercyjnej kasy.

6.7. Na stronie 12 Autor rozprawy ustawia w jeden szereg paradygmat obiektowy, service-oriented i agentowy, pisząc, że kolejny paradygmat jest wyższą formą nad poprzednim. Jest to płytka demagogia. Każdy z tych paradygmatów pasuje do innej klasy zadań i problemów. Aplikacje zorientowane na serwisy najczęściej wymagają projektowania i programowania przy pomocy technik obiektowych. Większość aplikacji biznesowych nie będzie używać ani SOA ani technologii agendowych, ponieważ do nich nie pasują. Technologie agentowe (jak pisałem wcześniej) jeszcze nie wyszły z fazy laboratoryjnej i nie wiadomo, czy przetrwają i w co się rozwiną. Analogią dla SOA w budownictwie są domy zrobione z wielkiej płyty, dzisiaj postrzeganej jako tandetna technologia budowlana. Jest prawdopodobne, że za 10 lat wszyscy zapomną o SOA, tak samo jak o innych technologiach „wielkopłytowych" („programming-in-the-large" z lat 1970-tych, „mega-programming" z lat 1980-tych, „component programming" z lat 1990). W istocie, są to konkurujące technologie i każda z nich może znaleźć swoją niszę zastosowań, ale na dzisiaj żadna z nich nie dominuje ani nie stanowi „kolejnej generacji".

6.8. Na stronach 12 i 13 znajduje się pożądana charakterystyka agentów. Jako człowiek stąpający po ziemi polemizowałbym z wieloma znajdującymi się tam postulatami. Przykładowo, jeżeli nie ma centralnego sterowania, to jak zapewnić bezpieczeństwo systemu? Jest to podstawowy wymóg wszystkich zastosowań biznesowych. Zatem brak centralnego sterownia jest względny: lokalnie, na określony czas agenci mogą nie podlegać centralnemu sterowaniu, ale ogólnie musi ono w jakiejś formie istnieć. Podobnie z warunkiem *adaptivity*. Większość zastosowań biznesowych tego nie będzie potrzebować, wiec jak uzasadnić dodatkowy koszt związanych z tą własnością? „Potencjalnym potencjałem", który być może się przyda w przyszłości? Zostanie on wyśmiany przez dowolne konsorcjum biznesowe. Generalnie, krytyczne zastosowania biznesowe nie będą tolerować cech (również u agentów), które zwiększają koszty nie zwiększając funkcjonalności, jakości lub pielęgnacyjności. Moim zdaniem, wymienione własności agentów są przykładem manowców, na które prowadzą

płytkie antropomorfizmy, czyli przypisaniu cech ludzkich zwykłemu kawałkowi kodu napisanego przez zwykłego programistę.

6.8. Na marginesie poprzedniej uwagi dodam, że w przedmiotowej rozprawie agenci są prostymi kawałkami kodu (słusznie) nie posiadającymi wymienionych na stronach 12 i 13 cech. Można mieć wątpliwości, czy warto nazywać ich „agentami", ale to już sprawa terminologii, w którą nie chcę wkraczać. Dobrze jest jednak wiedzieć, o czym mówimy i nie wkładać w termin zapożyczony z języka naturalnego elementów semantycznych z innego (rzeczywistego) świata. Jedynym celem tych agentów jest dokonanie obliczeń w sposób optymalny i najkrótszy, inne cele nie istnieją. Zatem są to „kawałki kodu" i nic więcej.

6.9. Rozdział 4 nie pasuje do tytułu rozprawy.

6.10. str.27-29. Obiektowość, jako informatyczna ideologia ma wiele sformułowań i definicji. Autor rozprawy przytacza jedną z nich, ale moje sformułowania pojęć obiektowości jest w wielu przypadkach całkowicie odmienne. (Patrz moje książki i strony WWW: http://www.sbql.pl/Topics/Object%20Oriented%20Concepts.html). Dotyczy to w szczególności takich pojęć jak klasa, enkapsulacja (hermetyzacja), komunikaty i polimorfizm, które definiuję inaczej. Nie mam oczywiście monopolu na poprawność definicji, ale moje definicje pasują do większości rozwiązań znanych z obiektowych języków programowania i baz danych, co nie zawsze ma miejsce w przypadku innych definicji.

6.11. str.43 i dalsze. Znajduje się tu opis agenta, jego cyklu życiowego, itd. W opisie tym brakuje mi kilku elementów, w szczególności:

- Jeżeli agent przenosi się w inne miejsce, to musi się przenosić również jego stan i środowisko wykonawcze (stosy, lokalne zmienne, trwałe zmienne). W wielu przypadkach takie przenosimy mogą być nieopłacalne, w szczególności jeżeli stan i środowisko wykonawcze jest dużej objętości. Przypuszczam, że dla celów obliczeń numerycznych jest to czynnik pomijalny, ale nie jest pomijalny np. dla środowisk obliczeniowych z bazą danych o ogromnych rozmiarach.
- Nie jest opisane explicite, w jaki sposób agenci reagują na polecenia ze strony podmiotu (agenta?) odpowiedzialnego za wykonanie całości zadania, w jaki sposób praca agentów jest synchronizowana (co jest ważne jeżeli kolejny agent ma skorzystać z wyników dostarczonych przez poprzedniego) oraz w jaki sposób przekazują wyniki swego działania do podmiotu odpowiedzialnego za całość.

- Czy agenci tworzą rozległą płaską strukturę, czy też mogą rekurencyjnie zagnieżdżać w sobie pod-agentów? (W moich pracach traktuję podobne twory jako zagnieżdżone.)

6.12. W pracy nieco brakowało mi konkretnych prostych przykładów. Wykład na poziomie rozważań ogólnych bardzo utrudnia zrozumienie motywacji wprowadzonych pojęć i formuł oraz ich rzeczywistego znaczenia.

6.13. Brak jest również jasności, które z wymienionych w pracy pomysłów i pojęć znalazły rzeczywiste odbicie w istniejących implementacja, a które są jedynie teoretyczną spekulacją.

6.14. str.63 i dalsze. Opisy przypadków są typu black box, brakuje szczegółów implementacji, przykładów i wyników.

6.15. str.72. Z podanej tabeli wynika, że prosty Round-robin jest jednak lepszy niż Diffusive scheduling. Na czym wiec polega przewaga tej ostatniej metody? Można się domyślać, że na znacznie łatwiejszym programowaniu, ale jeżeli wyniki wydajnościowe są jednak gorsze, to czy nie jest to zaprzeczenie głównego celu całości przedsięwzięcia?

Wiele bardziej drobnych uwag zaznaczyłem w dostarczonym mi egzemplarzu rozprawy (który mogę przekazać jej Autorowi). Podane uwagi krytyczne wynikają z obowiązku recenzenta i polemiki merytorycznej z tezami Autora. W żadnym stopniu nie powinny być traktowane jako podważające zasadnicze wyniki tej pracy, które oceniam bardzo wysoko.