Prof. dr hab. inż. Zbigniew J. Czech
Politechnika Śląska
Instytut Informatyki
e-mail: zczech@polsl.pl

Gliwice 25 lipiec 2018 r.

RECENZJA ROZPRAWY DOKTORSKIEJ DLA RADY NAUKOWEJ INSTYTUTU PODSTAW INFORMATYKI PAN

Tytuł rozprawy: Opracowanie w modelu PGAS wybranych, równoległych algorytmów grafowych i ich implementacja przy użyciu języka Java

Autorka rozprawy: mgr Magdalena Ryczkowska

1. **Teza (cele) rozprawy**

   We Wprowadzeniu do rozprawy Autorka nie sformułowała tezy, nie jest więc jasne co chciała w rozprawie wykazać. Stwierdziła natomiast, że celem rozprawy jest: (i) opracowanie w modelu PGAS wybranych, równoległych algorytmów grafowych, (ii) implementacja algorytmów w języku Java dla środowiska równoległego i rozproszonego, (iii) zbadanie wydajności i skalowalności zaimplementowanych algorytmów, w języku Java do przetwarzania grafów.

2. **Aktualność i ważność tematyki rozprawy**

   Nie ulega wątpliwości, że tematyka przedstawionej rozprawy jest aktualna i ważna. W obliczeniach równoległych rozważa się kilka modeli pamięci, Wśród nich można wymienić model jednolitej pamięci wspólnej (ang. *shared memory*) występujący w modelu obliczeń PRAM (Parallel Random Access Machine); model pamięci rozproszonej (ang. *distributed memory*) stosowany w obliczeniach rozproszonych; model rozproszonej pamięci wspólnej (ang. *distributed shared memory*, DSM). Odmianą modelu jednolitej pamięci wspólnej jest model pamięci o podzielonej, globalnej przestrzeni adresowej PGAS (Partitioned Global Address Space), użyteczny w obliczeniach wielowątkowych. Prace nad rozwijaniem modeli pamięci, w szczególności modelu PGAS, oraz implementowaniem programów w języku Java z zastosowaniem tego modelu są aktualne i wartościowe dla rozwoju dziedziny obliczeń równoległych.

3. **Wiedza Autorki oraz znajomość współczesnej literatury z dyscypliny naukowej, której dotyczy rozprawa**

   Autorka rozprawy wykazuje dobrą znajomość dorobku literaturowego dotyczącego modeli obliczeń równoległych, w szczególności modelu pamięci PGAS. Ma ugruntowaną wiedzę związaną z równoległymi obliczeniami wielowątkowymi implementowanymi w języku Java przy użyciu biblioteki PCJ (Parallel Computing in Java). Świadczą o tym treści zawarte w rozdz. 2–5.

   Wiedza Autorki dotycząca dziedzin algorytmiki i obliczeń równoległych jest niepełna. W obliczeniach sekwencyjnych używane są pojęcia złożoności obliczeniowej algorytmu, z wyróżnieniem złożoności czasowej $T(n)$ i pamięciowej $S(n)$, gdzie n jest rozmiarem danych wejściowych. W rozprawie Autorka mówiąc o złożoności czasowej używa określenia wydajność. Np. na rys. 3.5 wykresy złożoności czasowej (z osią rzędnych mierzoną w sekundach) opisane są jako „Testy wydajności ...” (podobnie jest na rys. 3.12, 3.14). Trudno jest akceptować, by wydajność była mierzona w sekundach. Rozważając złożoność pamięciową, Autorka używa określenia „narzut pamięciowy” (s. 67–68, 85).

   Jeśli chodzi o ocenę algorytmów równoległych, to Autorka nie podaje, że definiuje się w tym celu cztery wielkości charakterystyczne, zwane też metrykami (p oznacza liczbę procesorów lub liczbę rdzeni w procesorach wielordzeniowych): $T(p, n)$ – złożoność algorytmu równoległego, $S(p, n)$ – przyspieszenie, $C(p, n)$ – koszt równoległych obliczeń algorytmu,

$E(p, n)$ – **efektywność wykorzystania procesorów lub rdzeni. W rozprawie nie wyznacza się także na drodze teoretycznej, a także nie bada się eksperymentalnie, wielkości przyspieszenia algorytmu równoległego (w stosunku do algorytmu sekwencyjnego), kosztu i efektywności, np. dla zaproponowanego równoległego algorytmu BFS.**

4. **Ocena oryginalnego dorobku Autorki, jego znaczenie poznawcze oraz przydatność praktyczna dla nauki i techniki**

   Oryginalny dorobek Autorki zawarty jest w rozdziałach 3, 4 oraz 5 rozprawy.

   (a) **W rozdziale 3 Autorka prezentuje szczegóły wykonanych implementacji wybranych testów wzorcowych zawartych w witrynie Graph 500 (`http://graph500.org`) przy użyciu modelu PGAS, języka Java oraz biblioteki PCJ.**

   Zestaw testów wzorcowych Graph 500 składa się z trzech testów: Test 1 (nazwa oryginalna – Kernel 1) dotyczy konstruowania grafów, Test 2 (Kernel 2) dotyczy przeszukiwania grafu wszerz (ang. *Breadth-First Search*, BFS) oraz Test 3 (Kernel 3) dotyczy wyznaczania najkrótszych ścieżek z pojedynczego źródła (ang. *Single Source Shortest Paths*, SSSP).

   W rozprawie zaimplementowano testy wzorcowe Test 1 oraz 2, opisane, odpowiednio w podrozdz. 3.2 oraz 3.3. Test 3 został pominięty, nie jest jasne dlaczego. Autorka stwierdza jedynie (s. 101) „... że wersja 3.0.0 benchmarku Graph 500 wprowadza kilka zmian w stosunku do poprzedniej wersji ... jedną z nich jest dodanie nowego kernela ... znajdowania najkrótszych dróg w grafie z jednego źródła (SSSP)".

   Dla Testu 1 (podrozdz. 3.2): (i) Zrealizowano implementacje sekwencyjnych wersji tworzenia — na podstawie listy krawędzi otrzymanej z generatora grafów Kroneckera — reprezentacji grafów w postaci list sąsiedztwa wierzchołków oraz w postaci spakowanych wierszy (ang. *Compressed Sparse Row*, CSR). Dla reprezentacji grafów w obu postaciach wykonano pomiary czasów wykonania sekwencyjnej implementacji przeszukiwania BFS, z użyciem danych SCALE 22 i SCALE 24 (rys. 3.5). (ii) Zrealizowano kilka wersji równoległych implementacji tworzenia reprezentacji grafów w postaci spakowanych wierszy (CSR) (pkty 3.2.1 oraz 3.2.2). Przeprowadzono badania eksperymentalne wydajności wyznaczania reprezentacji CSR dla dwóch wersji implementacyjnych (wersja A i B; wersje te oznaczono także przez S1 i S2; nie jest jasne dlaczego zastosowano podwójne oznaczenia) z użyciem 32, 64 oraz 128 wątków dla danych SCALE 28 (rys. 3.18)

   Dla Testu 2 (podrozdz. 3.3): (iii) Przedstawiono koncepcje kilku wersji implementacji algorytmu przeszukiwania grafu wszerz (BFS). Pierwsza wersja oparta na synchronizacji wątków po zakończeniu przetwarzania wierzchołków na kolejnych poziomach przeszukiwania okazała się nieefektywna. Również druga rozpatrywana wersja równoległego algorytmu BFS, w której występowała wymiana dużej liczby krótkich wiadomości nie przyniosła zadowalających rezultatów (por. wyniki z rys. 3.24 uzyskane w klastrze Topola). Dopiero trzecia wersja przeszukiwania — oznaczona jako Algorytm 2 BFS (str. 91–92) — umożliwiła uzyskanie lepszej skalowalności i wydajności. Wersja ta została dostosowana do jednostronnej komunikacji oraz do modelu PGAS. W celu zmniejszenia kosztów komunikacji zastosowano metodę nakładania obliczeń i komunikacji, a także przyspieszono obliczenia przez użycie operacji bitowych.

   (b) **W rozdziale 4 Autorka przedstawia wyniki testów wydajnościowych dla algorytmów opisanych w rozdziale 3.**

   Testy zostały przeprowadzone z użyciem klastrów obliczeniowych Hydra (węzły istanbul, interlagos, magnycours), Topola (węzły haswell) i Okeanos. Szczegóły sprzętowe klastrów zostały omówione na s. 97–98. Jeśli chodzi o oprogramowanie, to użyto 64-bitowej Wirtualnej Maszyny Javy Oracle (Java 8) oraz biblioteki OpenMPI (wersja 1.6.5).

   Dla Testu 1 (pkty 4.4.1–4.4.2): (i) Przedstawiono wykresy wydajności TEPS (Traversed Edges Per Second) dla dwóch wersji S1 i S2 tworzenia reprezentacji grafów w

postaci spakowanych wierszy CSR, w porównaniu do wersji MPI pobranej z witryny Graph 500 i zaadaptowanej do środowiska, w którym przeprowadzono testowanie (por. rys. 4.4–4.7). (ii) Przeprowadzono badania zależności czasu wykonania Algorytmu 2 (BFS) mierzony w sekundach w zależności od rozmiaru bufora ACCUMULATE-BUFFER-SEND używanego w tym algorytmie (rys. 4.9–4.11). Parametrem na wykresach na tych rysunkach były liczby wątków wykonywanych w pojedynczym węźle obliczeniowym (16, 8 i 4 wątki). W wyniku badań, rozmiar bufora ustalono na 32 kB.

Dla Testu 2 (pkt. 4.4.3): (iii) Przedstawiono wykresy wydajności TEPS dla algorytmu BFS oraz grafów SCALE 26, 27 oraz 28 przy różnej liczbie wątków wykonywanych w węźle, tj. 64 wątki na węzeł zawierający cztery procesory po 16 rdzeni (rdzenie interlagos), oraz 4 wątki na węzeł zawierający dwa procesory zawierający po 6 rdzeni (rdzenie istanbul) (rys. 4.12). (iv) Na rys. 4.13 przedstawiono wykresy wydajności algorytmu BFS w porównaniu z wersją MPI dla grafu SCALE 26, przy różnych liczbach wątków na węzeł (w węzłach istabul, interlagos, magnycours). (v) Na rys. 4.14-4.15 przedstawiono czasy wykonania poszczególnych fragmentów algorytmu BFS. Autorka wyróżniła cztery fragmenty algorytmu, np. inicjacja struktur danych, oczekiwanie na dane i in. Wykresy wykonano dla danych SCALE 26 oraz węzłów istanbul i interlagos. Podobne wyniki, tylko na klastrze Okeanos są pokazane na rys. 4.17–4.18. (vi) Zaprezentowano wyniki wydajności algorytmu BFS (dla grafów SCALE 26 i 27) uzyskane na klastrze Hydra i Okeanos (rys. 4.16; należy je porównywać, jak sądzę, z wynikami z rys. 4.12). (vii) Porównano wyniki dla algorytmu BFS uzyskane na klastrze Okeanos dla implementacji PCJ oraz MPI (rys. 4.19). (viii) Opracowano wykresy pudełkowe czasów wykonania implementacji PCJ dla grafu SCALE 22 dla czerech i ośmiu wątków na węzeł w klastrze o węzłach zawierających dwa 24-rdzeniowe procesory (opis klastra jest na s. 122). Wykresy przedstawiono na rys. 4.21.

(c) W rozdziale 5 Autorka dokonała porównania wydajności implementacji algorytmu BFS w modelu PGAS z implementacją w modelu MapReduce, korzystając z platformy Hadoop.

Opracowane funkcje Map i Reduce zostały przedstawione na rys. 5.4–5.5. Porównanie czasów wykonania implementacji BFS-PCJ oraz MapReduce zaprezentowano na rys. 5.1–5.3.

Oceniając dorobek rozprawy stwierdzam, że ma on głównie charakter implementacyjno-eksperymentalny, natomiast strona teoretyczna rozprawy jest słaba.

Na dorobek implementacyjny oraz eksperymentalny rozprawy składają się implementacje napisane w języku Java przy użyciu biblioteki PCJ, wybranych testów wzorcowych z zestawu Graph 500. Opracowując implementacje Autorka stosowała model PGAS. Dla sporządzonych implementacji przeprowadziła dużą liczbę różnego rodzaju eksperymentów obliczeniowych, korzystając z wyposażenia sprzętowego dostępnych komputerów równoległych, oraz z ich środowisk programowych.

Uzyskane wyniki eksperymentalne mają znaczenie poznawcze umożliwiające ocenę przydatności języka Java oraz modelu PGAS w projektowaniu grafowych algorytmów równoległych, a także w ich praktycznej realizacji. Wyniki rozprawy stanowią cenny materiał źródłowy przydatny w dalszych badaniach dotyczących wymienionych zagadnień.

W odniesieniu do strony teoretycznej rozprawy, Autorka nie dokonała analizy złożoności czasowej sekwencyjnej wersji algorytmu przeszukiwania grafu wszerz (BFS). Jak wiadomo, złożoność tego algorytmu jest równa $T(n, e) = O(n+e)$, gdzie n jest liczbą wierzchołków grafu, a e liczbą jego krawędzi. Jeśli zachodzi $e = O(n)$ (dla grafów rzadkich), to algorytm ma złożoność liniową, gdy natomiast $e = O(n^2)$ (dla grafów gęstych), to ma złożoność kwadratową, względem liczby wierzchołków n. Tak więc, algorytm należy do grupy algorytmów wielomianowych – klasy P.

Wiadomo, że algorytm BFS jest P-zupełny (ang. P-*complete*), co oznacza, że prawdopodobnie jest trudny do zrównoleglenia (por. Greenlaw, R., H. J. Hoover, and W. L. Ruzzo. 1995. *Limits to Parallel Computation: P-Completeness Theory.* Oxford: Oxford University Press, Part II, pp. 146–147. www.cs.armstrong.edu/greenlaw/research/PARALLEL).

Wielomianowe algorytmy efektywnie rozwiązywane równolegle (np. sortowanie, mnożenie macierzy) należą do klasy NC (ang. *Nick's Class*). Dla wielomianowej liczby procesorów, $p = n^{O(1)}$, można je rozwiązać w czasie wielomianowo-logarytmicznym, $O(\log^k n)$, dla stałych $k > 0$. Można udowodnić, że NC $\subseteq$ P, choć problem NC $\neq$ P? jest otwarty.

Tak więc, P-zupełny problem przeszukiwania grafu wszerz mimo użycia dowolnej liczby procesorów do jego zrównoleglenia pozostanie problemem wielomianowym, przy wielce prawdopodobnym założeniu, że NC $\neq$ P.

I tu rodzi się pytanie: Jak biorąc pod uwagę powyższe trudności należy rozumieć konkluzję Autorki zamieszczoną w Podsumowaniu rozprawy, że „Przedstawione w pracy wyniki testów wydajnościowych pokazują, że obliczenia równoległe i rozproszone na danych grafowych wykonane w modelu PGAS w języku Java z użyciem biblioteki PCJ, pozwalają na uzyskanie dobrej wydajności i skalowalności.".

5. Wady i słabe strony rozprawy, uwagi dyskusyjne

(a) Jedną z cech algorytmów badanych w rozprawie jest skalowalność. Autorka definiuje mierzalność tej cechy dwojako (s. 97):

A. „Skalowalność algorytmów była mierzona głównie dla stałego rozmiaru danych wejściowych (ang. *strong scaling* – czyli badanie zmiany czasu wykonania wraz z liczbą wątków dla stałego rozmiaru problemu). Jeśli nie podano inaczej wykresy przedstawiają silną skalowalność."

B. „Przy słabej skalowalności (ang. *weak scaling*) bada się zmianę czasu wykonania wraz z liczbą wątków dla stałego rozmiaru problemu przypadającego na jeden wątek. Przy badaniu słabej skalowalności, zwiększając liczbę wątków, rozmiar wejściowego problemu również rośnie."

Definicje te nie są ścisłe z dwóch powodów. Po pierwsze, mówi się w nich o czasach wykonania [algorytmów], podczas gdy w rozprawie, komentując skalowalność Autorka odnosi się do wykresów wydajności TEPS, która nie jest mierzona za pomocą jednostek czasu, tylko liczbą odwiedzonych (ang. *traversed*) krawędzi w ciągu sekundy (por. np. s. 88, s. 103 oraz rys. 4.4-4.6, s. 110 oraz rys. 4.12–4.13 i in).

Po drugie, „... zmiana czasu wykonania wraz z liczbą wątków ..." określa, jak należy sądzić, szybkość narastania wykresu TEPS. Innymi słowy, Czy miarą skalowalności jest szybkość narastania wykresu? albo inaczej: Czy to oznacza, że powinniśmy oceniać szybkość narastanie wykresu i oceniać, że skalowalność jest np. zła, średnia, dobra, bardzo dobra?

Gwoli uzupełnienia dodajmy co następuje. Klasyczna definicja skalowalności mówi, że system równoległy (tj. para algorytm–komputer równoległy) jest skalowalny, jeśli jego efektywność może być utrzymywana na zadanym, stałym poziomie (z przedziału $[0.0, 1.0]$) przez jednoczesne zwiększanie liczby procesorów (lub rdzeni) i rozmiaru rozwiązywanego problemu. Skalowalność systemu równoległego ocenia się na podstawie szybkości wzrostu (w sensie rzędu) funkcji izoefektywności, która określa rozmiar problemu n jako funkcję liczby procesorów p (przykładem może być funkcja $n(p) = \Theta(p \log p)$). Jeśli rozmiar rozwiązywanego problemu będziemy zmieniać zgodnie z tą funkcją, zwiększając jednocześnie liczbę procesorów p, to zapewnimy stałą, zadaną efektywność wykorzystania procesorów. Oczywiście, dla pewnej liczby procesorów p, dalsza skalowalność systemu nie będzie możliwa (czyli nie zapewnimy stałej efektywności), ponieważ rozmiar problemu określony przez $n(p)$ będzie większy od rozmiaru pamięci, którymi dysponują procesory/rdzenie danego komputera równoległego.

(b) Istotną wadą rozprawy jest brak badań zaprojektowanych równoległych algorytmów pod względem przyspieszenia, kosztu obliczeń i efektywności wykorzystania procesorów (o czym wspomniałem w pkt 3 niniejszej recenzji).

Badanie przyspieszenia dla zaproponowanego algorytmu BFS umożliwiłoby ocenę, czy algorytm równoległy wyznacza drzewo przeszukiwania wszerz szybciej niż algorytm sekwencyjny, biorąc pod uwagę P-zupełność problemu BFS.

We współczesnej algorytmice, zarówno sekwencyjnej, jak i równoległej, rozważa się nie tylko to, czy określony problem da się rozwiązać, ale także jaka jest minimalna ilość zasobów, której trzeba użyć, aby znaleźć poszukiwane rozwiązanie. W obliczeniach równoległych wyznaczamy wówczas koszt obliczeń równoległych zdefiniowany jako $C(p,n) = pT(p,n)$ będący iloczynem liczby p użytych procesorów/rdzeni oraz złożoności algorytmu równoległego $T(p,n)$, która określa liczbę kroków obliczeniowych (operacji elementarnych), którą wykonał każdy z tych procesorów/rdzeni. Niestety, w rozprawie analiza teoretyczna i badanie eksperymentalne kosztu obliczeń zaproponowanego algorytmu BFS zostały pominięte.

Podobnie pominięte zostały badania efektywności wykorzystania procesorów/rdzeni. Jeśli efektywność jest mała, to koszt obliczeń równoległych jest duży. A to oznacza, że „marnujemy" możliwości zasobów w postaci procesorów/rdzeni, za pomocą których rozwiązujemy dany problem.

(c) **W Podsumowaniu rozprawy Autorka wyciąga następujący wniosek:**

> **„Opracowane rozwiązania oraz uzyskane wyniki świadczą o możliwym wykorzystaniu modelu PGAS i języka Java do efektywnego przetwarzania danych grafowych przy analizie danych (...)."**

Mam wątpliwości, czy wniosek jest uzasadniony, jako że efektywność implementacji Java-PCJ dla większych danych (SCALE 28, rys. 4.19) jest wyraźnie gorsza od implementacji C-MPI.

Na marginesie, w rozprawie efektywność i skalowalność algorytmów jest badana dla grafów o rozmiarach SCALE 22–28, które w zestawie testów wzorcowych Graph 500 są klasyfikowane jako „Toy problems". Badania dla problemów Mini (SCALE 29), Small (SCALE 32), ..., Huge (SCALE 42) umożliwiłyby dokładniejszą ocenę efektywności implementacji Java-PCJ.

Rozpatrując skalowalność, należy brać pod uwagę to, że Java jest językiem interpretowanym. Szybkość interpretowania kodu bajtowego przez maszynę JVM (nawet uwzględniając, że jest on kompilowany do kodu maszynowego w trakcie działania programu) jest mniejsza niż szybkość wykonywania kodu maszynowego programu w języku C tworzonego przez kompilator. Ponadto zauważmy, że w programie BFS tzw. część „operacyjna" — polegająca jedynie na odwiedzaniu krawędzi i wierzchołków grafu — jest znikoma. Sytuacja zmieni się, jeśli będziemy rozwiązywać bardziej złożone problemy, o rozbudowanej części „operacyjnej", np. badanie planarności grafu, kolorowania grafu, etc. Wówczas nawet stosunkowo niewielkie spowolnienie szybkości wykonywania kodu Javy wpłynie ujemnie na skalowalność (łatwo to sprawdzić mierząc czas wykonania wybranego, nieco bardziej złożonego algorytmu grafowego zaimplementowanego w językach Java i C).

(d) **Jak powszechnie wiadomo, model obliczeń PRAM jest nierealizowalny technicznie, ponieważ jest nieskalowalny. Przy wzroście liczby procesorów (np. rzędu milionów) nie można bowiem zapewnić stałego czasu dostępu (ang. *random access*) do dowolnej komórki jednolitej pamięci wspólnej. Czytając rozprawę, szukałem odpowiedzi na pytanie, czy model PGAS jest skalowalny, ale nie znalazłem uwag na ten temat. Sądzę, że model PGAS też nie jest skalowalny, ponieważ fizycznie pamięć modelu jest rozproszona. Składają się na nią pamięci wspólne rdzeni poszczególnych procesorów wchodzące w skład węzłów obliczeniowych. A te komunikują się ze sobą za pomocą sieci połączeń o określonej konfiguracji oraz skończonej przepustowości. W rezultacie, globalna przestrzeń adresowa modelu PGAS nie składa się z komórek o swobodnym dostępie, tj. o tym samym czasie dostępu niezależnie od adresu komórki.**

(e) **Oś rzędnych na rys. 4.8 jest opisana „Przyspieszenie". O jakie przyspieszenie chodzi? W definicji słabej skalowalności (s. 97) mówi się o badaniu zmiany czasu wykonania (...).**

(f) **Cytat na s. 13: „Dziś większość procesorów działa z szybkością do 4 GHz ... Obecnie jednak technologia doszła do granicy możliwości zwiększania częstotliwości taktowania procesora" – Oto jeden z kilku cytatów w Sieci: „10 Ghz processors already exist. It is just that they are inefficient heat spewing furnaces, so they are not used in**

everyday applications. Watch the AMD team run a unlocked processor at 8.49 GHz (after cooling it with liquid nitrogen, no less. With better cooling tech and changes at the drawing board, it can be made to run at even 10+ GHz)"

(g) W rozprawie jest sporo błędów językowych, błędów korekty, określeń żargonowych, a także innych uchybień. Poniżej przedstawiam niektóre z nich:

i. błędne pojęcia: szybkości taktowania zegara ... działa z szybkością do 4 GHz – w GHz mierzymy częstotliwość taktowania zegara, a nie szybkość (s. 13); wydajność biblioteki – pytanie: jak się mierzy wydajność biblioteki będącej zbiorem funkcji/procedur (s. 18).

ii. personifikacje: Programowanie równoległe dostarcza (s. 14), systemy oferują (s. 16), Język Java wykorzystywał (s. 17), Przedstawiony sposób zakłada (s. 78), wyniki dla PCJ zachowują się (s. 87).

iii. błędy językowe: ilości wierzchołków, Graph 500 posiada, wątek 1 posiada (s. 68), oparta o przesunięcia (s. 70), ilości węzłów (s. 75), ilość przesyłanych wiadomości (s. 86), aby nie nadpisać (s. 86), zakomentowane definicje (s. 100), zedytować (s. 100), przeprocesowanie odebranych (s. 111).

iv. niejednolita/niespójna terminologia: za pomocą testu wzorcowego (ang. benchmark) ... Benchmark reprezentuje (s. 16), na różnych benchmarkach (s. 17), otrzymanych z Generatora Benchmarku Graph500 (s. 18) [jeśli ma to być neologizm w języku polskim, to powinien być pisany jako benczmark], dla testu Kernela 1 (s. 20).

v. niejednolita/niespójna terminologia c.d.: nieprzerwany wzrost wydajności JVM np. garbage collector'a ... automatycznego zwalniania pamięci ... automatyczne odśmiecanie pamięci (s. 17).

vi. określenia żargonowe: za pomocą *wrapperów* (s. 100), dodanie nowego kernela (s. 101).

vii. błędy interpunkcji: narzędzi których → narzędzi, których (s. 17), w odpowiednim formacie dla którego → w odpowiednim formacie, dla którego (s. 19).

viii. błędy korekty: sklowalności (s. 18), $N = 2^SCALE$ (s. 64), do liczny wierzchołków, głownie (s. 68), opis algorytmy (s. 85), równica (s. 113).

ix. inne: Keneth'a Appel'a → Kenetha Appela (s. 13).

x. błędy redakcji: We Wprowadzeniu do rozprawy Autorka formułuje cztery cele (s. 18). A następnie (ciągle we Wprowadzeniu) komentuje stopień ich realizacji. Np. w odniesieniu do celu: „opracowanie w modelu PGAS ...", Autorka napisała: „W ramach rozprawy sporządzono równoległe wersje ... w repozytorium Git." Taki „podsumowujący" komentarz powinien się znaleźć w Podsumowaniu a nie we Wprowadzeniu do rozprawy – i co ciekawe on tam jest *w identycznym brzmieniu*! (s. 125). Podobna uwaga dotyczy niepotrzebnego powielania tekstów „podsumowujących" cele rozprawy (por. fragmenty na s. 18, 20 z fragmentami na s. 125). Czemu mają służyć takie powielania tekstów?
Ponadto, czy szczegółowy opis zrealizowanych pakietów w języku Java powinien się znaleźć we Wprowadzeniu do rozprawy (s. 18–20)?

6. Wniosek końcowy

Biorąc pod uwagę zalety i wady rezultatów badawczych przedstawionych w rozprawie stwierdzam, że rozprawa doktorska mgr Magdaleny Ryczkowskiej spełnia w stopniu dostatecznym wymagania obowiązujących przepisów dotyczących zawartości i formy rozpraw doktorskich. Wnoszę zatem o dopuszczenie mgr Magdaleny Ryczkowskiej do następnej fazy przewodu doktorskiego.

Na podkreślenie zasługuje opublikowanie przez Autorkę części wyników rozprawy w siedmiu sprawozdaniach konferencyjnych. Brak jest publikacji w czasopismach z listy JCR.

J. Cisch