Warszawa, 17.03.2020

dr hab. Marcin Paprzycki
Instytut Badań Systemowych PAN
ul. Newelska 6
01-447 Warszawa

Recenzja Osiągnięć Naukowych

Dr Zbigniewa Marszałka
Przygotowana w związku z postępowaniem habilitacyjnym

Przewód habilitacyjny Pana dr Zbigniewa Marszałka jest prowadzony na Wydziale Inżynierii Mechanicznej i Informatyki Politechniki Częstochowskiej, w dziedzinie Nauk Technicznych, w dyscyplinie Informatyka Techniczna i Telekomunikacja. Postępowanie to zostało wszczęte przez Radę Doskonałości Naukowej w dniu 28 października 2019.

1. Charakterystyka habilitanta

Pan dr Zbigniew Marszałek ukończył, w roku 1978, studia na Wydziale Matematyki Fizyki i Chemii, Uniwersytetu Śląskiego, uzyskując tytuł magistra matematyki. Stopień naukowy doktora nauk matematycznych uzyskał w roku 1986, na Politechnice Śląskiej na Wydziale Matematyczno Fizycznym, na podstawie doktoratu pt. „O pewnych rezydualnych metodach rozwiązywania układów równań liniowych". Od roku 1978 jest on zatrudniony na Politechnice Śląskiej, kolejno na stanowiskach: Asystent, Adiunkta, Wykładowcy i Starszego Wykładowcy. Z przedstawionej przez Kandydata dokumentacji wynika, że nie ubiegał się on do tej pory o nadanie stopnia doktora habilitowanego.

2. Ocena jednotematycznego cyklu publikacji

Dr Zbigniew Marszałek przedstawił do oceny jednotematyczny cykl publikacji zatytułowany: „Nowe równoległe metody sortowania przez scalanie dużych zbiorów danych w bazach NoSQL". Cykl ten składa się z 14 publikacji. W szczególności zawiera on 2 monografie (współautorskie) wydane przez Wydawnictwo Politechniki Śląskiej, 4 artykuły w czasopismach zagranicznych (wydawnictwa: Hindawi, MDPI, JUCS, oraz Kaunas University of Technology), 2 artykuły w czasopismach krajowych (Wydawnictwo Uniwersytetu Warmińsko-Mazurskiego), oraz 6 artykułów opublikowanych w materiałach konferencyjnych (4 z nich wydano w materiałach corocznej konferencji organizowanej przez Kaunas University of Technology, 1 w materiałach konferencji ICAISC, oraz 1 w materiałach konferencji KICSS). Wszystkie publikacje w materiałach konferencyjnych zostały opublikowane w różnych seriach wydawniczych wydawnictwa Springer.

2.1 Tytuł cyklu publikacji

Zanim przejdę do oceny poszczególnych publikacji, ich grup, i całego cyklu, chciałbym zwrócić uwagę na tytuł całego cyklu, który jest mylący. Po pierwsze, związek pomiędzy przedstawionymi wynikami a bazami danych NoSQL nie został jasno przedstawiony. W poszczególnych pracach pojawiają się nawiązania do faktu, ze sortowanie jest ważne w bazach danych NoSQL. Ponadto znajdujemy (podobne do siebie) rysunki przedstawiające proste procesy, w których to procesach „występuje ikonka

bazy danych". Jednakże poza tym nie ma, pogłębionego opisu powiązania pomiędzy omawianymi metodami sortowania a systemami bazodanowymi. Istnienie bazy danych jako elementu systemu, w którym ma miejsce sortowanie, nie jest uwzględniane ani w modelach formalnych złożoności modelowanych procesów, ani w częściach eksperymentalnych ocenianych publikacji. Ten ostatni fakt jest szczególnie istotny i wymaga podkreślenia. *W części eksperymentalnej żadnej pracy nie występuje „uruchomiona baza danych, będąca częścią przeprowadzonych eksperymentów"*. Zasadniczo rzecz biorąc, z przedstawionych do oceny prac można byłoby usunąć materiał dotyczący baz NoSQL, lub zastąpić je bazami relacyjnymi (gdzie także mamy do czynienia z sortowaniem – na przykład w procesie indeksowania lub re-indeksowania bazy) i zawartość merytoryczna pozostałych części prac nie musiałaby w żaden sposób ulec zmianie. Innymi słowy *komponent bazodanowy zawarty w tytule jest całkowicie zbędny i nic nie wnosi.*

Po drugie, stwierdzenie, że prowadzone badania mają cokolwiek wspólnego z dużymi danymi (ang. Big Data) nie ma żadnych podstaw. Największe zbiory danych, które zostały wykorzystane w przeprowadzanych eksperymentach to 100,000,000 elementów. Zakładając, że są to liczby rzeczywiste 16 bitowe (32 bitowe), to jest to ~1.6 (~3,2) GByte danych. Biorąc pod uwagę, że w chwili obecnej dowolny telefon komórkowy (model podstawowy) ma co najmniej 3 GByte pamięci, a podstawowe modele laptopów mają co najmniej 8 GByte pamięci, to dane które były rozważane w pracach Kandydata zdecydowanie nie odpowiadają nawet najbardziej elastycznemu znaczeniu pojęcia duże dane. Należy tutaj podkreślić, że określenie duże dane rozumiane powinno być jako: „duże, zmienne i różnorodne zbiory danych, których przetwarzanie jest trudne w danym momencie czasowym" (MP: na podstawie: Komisja Europejska, materiały DG Connect; zgodne z innymi źródłami). W tym kontekście, w przedstawionych pracach nie znajdziemy żadnych stwierdzeń, które by sugerowały, że przetwarzane dane były „zmienne" lub „różnorodne". Ponadto nie znajdziemy również sugestii, że danych „było tak dużo", że ich przetwarzanie było w jakikolwiek sposób „trudne" (na przykład dane nie mieściły się w pamięci wykorzystywanego komputera). Tak więc *żadna z przedstawionych do oceny prac nie dotyczy dużych danych.*

Z powyższych rozważań wynika, że ***przedstawiony do oceny cykl publikacji nie stanowi wkładu w obszarze „nie-relacyjnych baz danych" ani w obszarze „przetwarzania dużych zbiorów danych"***. Oznacza to również, że przedstawiony do oceny cykl publikacji dotyczy „Nowych równoległych metod sortowania przez scalanie". Powyższa analiza i otrzymany wniosek będą istotne w dalszej części niniejszej recenzji.

2.2. Analiza poszczególnych tekstów

Przejdziemy teraz do oceny zawartości merytorycznej tekstów należących do, przedstawionego do oceny, jednolitego cyklu publikacji.

2.2.1 Ulepszanie algorytmu QuickSort

Analizę indywidualnych tekstów rozpocznę od kuriozalnego artykułu „Preprocessing Large Data Sets by the Use of Quick Sort Algorithm". Istotą tego tekstu jest zaproponowanie by w przypadku algorytmu QuickSort wykorzystać losowy wybór elementu środkowego (ang. pivot). Problem polega na tym, że podejście to było znane znacznie wcześniej. Wystarczy się bowiem zapoznać z klasycznym podręcznikiem: T.H. Cormen, C.E. Leisserson, R.L. Rivest, C. Stein, Introduction to Algorithms, Second Edition, McGraw Hill, 2001, Sekcja 7.3, strony 153-154. Ponadto należy przywołać publikację: Aminu Mohammed and Mohamed Othman, Comparative Analysis of Some Pivot Selection Schemes for Quicksort Algorithm, Information Technology Journal, 2007, 6: 424-427, w której przeprowadzono eksperymentalne porównanie siedmiu różnych metod wyboru elementu środkowego (w tym różnych wariacji wykorzystujących element losowy). W tymże artykule możemy przeczytać: „Using a random

partitioning element will virtually prevent the anomalous cases from happening in practical sorting situations, but random number generation can be relatively expensive and does not reduce the average running time of the rest of the algorithm (Singleton, 1969)". Oznacza to, że metoda, która została „opracowana" przez Kandydata, była rozważana już w roku 1969. Jednoznacznym jest więc, że omawiany artykuł ***nie ma żadnego znaczenia naukowego***. Warto zauważyć, że w ciekawym świetle stawia to również pracę numer 7, która nie jest częścią cyklu publikacji podlegających ocenie, która również omawia randomized QuickSort. Ponadto, zauważyć należy, że tekst ten nie ma nic wspólnego z programowaniem równoległym, nie jest więc jasnym dlaczego został on włączony do jednolitego cyklu publikacji dotyczących programowania równoległego.

2.2.2 Teksty wydane w czasopismach i materiałach konferencyjnych które nie zawierają elementów programowania równoległego

Weźmy teraz pod uwagę następujące 4 artykuły:
a) Modified Merge Sort Algorithm for Large Data Sets
b) Performance Tests on Merge Sort and Recursive Merge Sort of Big Data Processing
c) Performance Test on Triple Heap Sort Algorithm
d) Novel Recursive Fast Sort Algorithm

Po pierwsze, ***żaden z nich nie dotyczy programowania równoległego***. Nie jest więc jasnym dlaczego zostały włączone do ocenianego jednolitego cyklu publikacji dotyczących „Nowych równoległych metod sortowania przez scalanie".

Po drugie, porównywanie metod rekurencyjnych i iteracyjnych (nie rekurencyjnych) jest być może ciekawym ćwiczeniem dla studentów przedmiotu „Algorytmy i złożoność", ale zdecydowanie nie stanowi wkładu naukowego w obszarze metod sortowania. Warto tutaj zwrócić uwagę na fakt, że porównywanie wybranych aspektów rekurencyjnych i nie-rekurencyjnych podejść do sortowania można znaleźć na wielu stronach w Internecie, łącznie z kodami obu podejść przedstawionymi dla w różnych języko programowania (zawierają je portale takie jak, na przykład, GeeksforGeeks). Tak więc można stwierdzić, że *artykuły porównujące iteracyjne i rekurencyjne implementacje metod sortowania są przykładem eksperymentalnego wykazywania oczywistego i nie należą do osiągnięć jakich oczekuje się w kontekście habilitacji.*

Po trzecie, *zaproponowane metody ulepszania wybranych metod sortowania, na przykład przejście z „dwu-scalania" do „trzy-scalania" stanowią niewątpliwie pewien wkład w praktykę metod sortowania, ale jest to raczej przyczynek niż wkład znaczący.*

Podsumowując wspólnie powyższe cztery teksty można stwierdzić jednoznacznie, że ***nie wnoszą one żadnego wkładu w teorię i praktykę programowania równoległego a ich wkład w obszarze sortowania jest marginalny***.

2.2.3 Dwie monografie

W skład jednolitego cyklu publikacji zostały włączone dwie, współautorskie, monografie wydane przez Wydawnictwo Politechniki Śląskiej. Ponownie, wbrew ich tytułom, twierdzenie, że ich zawartość dotyczy dużych danych jest mylne, albowiem wszystkie eksperymenty wykonane są dla nie więcej niż 100,000,000 danych. Nie są to ilości danych, dla których występowała by jakaś szczególna trudność realizacji procesu sortowania związana z rozmiarem (nawet w czasie kiedy monografie były przygotowywane, czyli w latach 2013 i 2014; patrz, również, poniżej). Warto również zauważyć, że tak w monografiach jak i w innych publikacjach praktycznie nie rozważa się wpływu typu sortowanych danych na algorytmy sortujące.

Ponadto monografie nie zawierają materiału związanego z programowaniem równoległym a związki omawianego materiału z bazami NoSQL są bardzo powierzchowne. *Tak więc wyniki przedstawione w obu monografiach nie stanowią wkładu w rozwój wiedzy dotyczącej: baz danych, dużych danych, programowania równoległego.*

Wkład monografii w obszar nauki dotyczący sortowania polega na zebraniu i usystematyzowaniu wiedzy, bardziej niż jej rozszerzeniu – chociaż z pewnym rozszerzeniem mamy niewątpliwie do czynienia. Analizując zawartość merytoryczną obu monografii można stwierdzić, że stanowiłyby one bardzo dobrą literaturę uzupełniającą dla przedmiotu: Algorytmy i złożoność obliczeniowa. ***Nie jest to jednak istotny wkład naukowy w dziedzinę informatyki technicznej i telekomunikacji, a bardziej wkład w obszarze dydaktyki informatyki.***

2.2.4 Teksty dotyczące programowania równoległego

Zawartość merytoryczna pozostałych siedmiu tekstów dotyczy rzeczywiście programowania równoległego. Są one więc koncepcyjnie zgodne z zaproponowanym tytułem jednolitego cyklu publikacji. Spośród nich chciałbym wyróżnić teksty opublikowane w dwu czasopismach: J.UCS oraz Journal of Information Technology and Control, które omówię osobno. Na podstawie analizy pozostałych 5 tekstów doszedłem do następujących wniosków:

(i) Zrównoleglanie algorytmów sortowania jest zagadnieniem, które było badane od dawna. Jednakże w kontekście prac Kandydata warto zwrócić uwagę, na przykład, na materiał z roku 2014, który znajdziemy na stronie: https://www.drdobbs.com/parallel/parallel-in-place-merge-sort/240169094. Są tam przedstawione eksperymentalne wyniki sortowania równoległego dla systemu z procesorem Intel i7 3630QM „quad-core CPU with hyperthreading, running at 3.2 GHz, using 16 GB memory" (warto tutaj zauważyć, że w pracach Kandydata regularnie pomijane są szczegółowe informacje o komputerze, na którym wykonano eksperymenty – np. takie jak dostępna pamięć, oraz w jaki sposób mając do dyspozycji procesor o 4 rdzeniach wykonano eksperymenty dla 8 procesorów).

Jak widać z wyników zawartych na ww. stronie, w roku 2014 przeprowadzone zostały eksperymenty dla 1,000,000,0000 elementów (czyli z 10x większymi danymi niż największe eksperymenty raportowane w pracach Kandydata). Przedstawiona na tejże stronie WWW Tabela 1 zawiera wyniki znacznie bardziej wszechstronne (z punktu widzenia oceny efektywności różnych podejść do realizacji równoległych algorytmów sortowania) od tych, które znajdują się w pracach Kandydata. Natomiast Tabela 2 (na następnej – podłączonej – stronie WWW) przedstawia wszechstronne wyniki otrzymane po optymalizacji zastosowanych podejść. Jest to o tyle ważne, że w pracach Kandydata nie zostało uwzględnione to w jaki sposób można zoptymalizować kod działający na komputerze równoległym, biorąc pod uwagę jego charakterystyki sprzętowe. ***Bez takich rozważań***, których przykłady istnieją (jak widać z powyższej referencji) w Internecie i pochodzą z czasów zanim jeszcze Kandydat zaczął prowadzić badania dotyczące równoległych metod sortowania oznacza, że ***wkład prac Kandydata w obszarze praktycznych aspektów programowania równoległego jest minimalny***.

Warto tutaj również wspomnieć, że równoległa realizacja sortowania jest od dawna przedmiotem nauczania studentów studiów licencjackich; na przykład była ona rozważana w ramach przedmiotu CME 323: Distributed Algorithms and Optimization, oferowanego na Stanfordzie wiosną 2015. Ten przedmiot jest tylko przykładem, albowiem implementacja i eksperymentowanie z równoległym sortowaniem pojawiało się jako proponowany element studiów licencjackich już znacznie wcześniej (na przykład zostało to zaproponowane w opublikowanej w roku 1994 pracy: "Teaching Parallel Computing without a Separate Course," M. Paprzycki, J. Zalewski, Nevison, C.H. (ed.), Proceedings of the Conference on Parallel Computing for Undergraduates, Colgate University, June, 1994, 18/1-19).

(ii) Podchodząc całościowo do ***wyników*** zawartych w ocenianych pracach, należy zauważyć, że nawet w latach kiedy teksty te były publikowane, ***nie wnosiły one istotnego wkładu w obszar***

programowania równoległego. Należy tutaj podkreślić, że *teksty te miałyby pewną wartość naukową w latach 1985-1995*, gdy wyprodukowane zostały pierwsze komercyjne komputery równoległe o pamięci współdzielonej (ang. shared memory). Komputery te (produkowane przez firmy takie jak Sequent, Alliant, Pyramid czy Encore) miały, zwykle, 8-16 procesorów (tylko czasami 20 lub 32). Wówczas miały również miejsce pierwsze eksperymenty z programowaniem równoległym, realizowanym na rzeczywistych, komercyjnych, komputerach. Przykładem takich badań są wyniki opublikowane w pracy: R. S. Francis and I. D. Mathieson, "A benchmark parallel sort for shared memory multiprocessors," IEEE Transactions on Computers, vol. 37, no. 12, 1619-1626, 1988, gdzie przedstawione zostały wyniki eksperymentów z algorytmem merge sort na 12 procesorowym komputerze Sequent Balance 21000. W tamtych latach badania naukowe Kandydata, i eksperymenty przeprowadzone na nie więcej niż 8 procesorach miałyby sens i byłyby ciekawe. Wówczas (przed rokiem 1995) nawet rozmiary danych, na których zostały przeprowadzone eksperymenty opisywane w ocenianych publikacjach, były rzeczywiście dużymi danymi.

(iii) Po roku 1995 bardzo wiele się zmieniło jeśli chodzi o programowanie równoległe. (1) Komputery z pamięcią współdzieloną „uległy miniaturyzacji" i stały się pojedynczymi procesorami wchodzącymi w skład większych systemów komputerowych, np. już około roku 2006 w Polsce można było bez problemu nabyć serwery, które miały dwa procesory na jednej płycie, każdy z nich był procesorem dwurdzeniowym, a każdy rdzeń miał wiele wątków. Przykładem procesora, który jest „mini-komputerem o pamięci współdzielonej", jest również procesor AMD Opteron 8356 8p na którym Kandydat przeprowadził eksperymenty opisane w pracy „Parallelization of Fast Sort Algorithm" (z podobnymi procesorami również mamy do czynienia w pozostałych pracach Kandydata). Wprawdzie znane są również procesory z linii rozwojowej Intel Xeon Phi, które miały 68 rdzeni, ale (2) programowanie równoległe, jako dziedzina badań, skupiło się na komputerach o pamięci rozproszonej (ang. distributed memory). Dotyczyło to nie tylko superkomputerów (takich jak Intel Hypercube, którego pierwsza wersja powstała w roku 1985 i który miał do 128 procesorów; nie mówiąc już o maszynie Intel Paragon, która w latach 90tych ubiegłego wieku miała nawet 4096 procesorów), jak i małych maszyn (takich jak klastry typu Beowulf, które zwykle miały do 32 procesorów – w zależności od technologii służącej do łączenia maszyn w klastrze). W pracach Kandydata nie ma żadnych informacji na temat realizacji rozważanych algorytmów na komputerach o pamięci rozproszonej. Nie ma też żadnych wyników dla więcej niż 8 procesorów (co należy porównać z dostępną już od roku 1985 liczbą procesorów). (3) W przedstawionym do oceny zestawie publikacji, nie ma też rozważań dotyczących realizacji sortowania na komputerach, które składają się z wielu procesorów wielordzeniowych, z których każdy rdzeń jest wielowątkowy.

Należy tutaj podkreślić, że nie jest żadnym argumentem brak dostępu (w Polsce) do sprzętu z dużą liczbą procesorów. W okresie kiedy były prowadzone badania, opisywane w ocenianych pracach, dostępne były dla polskich naukowców „duże maszyny" tak w Gdańsku, jak i w Warszawie (nie wspominając o innych ośrodkach naukowych). Można więc było przetestować proponowane podejście dla dużych danych i dużej liczby procesorów. Brak rozważań dotyczących komputerów o pamięci rozproszonej i dla więcej niż 8 procesorów sprawia, że ***omawiane prace nie wnoszą wartościowego wkładu w obszar programowania równoległego*** (i nie wnosiły takiego wkładu również w momencie publikacji).

(iv) W pracach kandydata pojawiają się teoretyczne modele obliczeń równoległych. Na przykład w pracy „Modification of Parallelization for Fast Sort Algorithm", w Sekcji 2, wprowadzone zostają model PRAM, i jego wersje: CRCW, ERCW, CREW i EREW. Jednakże tak w tym artykule (poza Sekcją 2), jak i w innych tekstach gdzie modele te zostają wymienione, nie są one w sposób istotny wykorzystane. Pojawia się wprawdzie sugestia, że modele te były brane pod uwagę przy opracowaniu koncepcji zrównoleglenia algorytmów sortujących, ale informacja ta nie została w wystarczający sposób rozwinięta w dalszej części tej publikacji (jak i pozostałych, przywołujących te modele).

Warto tutaj zauważyć, że powyższe modele były powszechnie wykorzystywane w badaniach nad programowaniem równoległym mniej więcej do końca ubiegłego stulecia. Następnie ich popularność gwałtownie zmalała i w chwili obecnej bardzo rzadko występują w literaturze przedmiotu. Wynika to, miezy innymi, z faktu, że nie były one w stanie właściwie (na koniecznym poziomie granularności) uchwycić złożoności współczesnych komputerów równoległych. Jeśli weźmiemy pod uwagę, że współczesne systemy, nawet średniej wielkości, składają się z procesorów, które mają wiele rdzeni (nawet telefony komórkowe mają 8-rdzeniowe procesory), każdy rdzeń ma wiele wątków (tak zmiennoprzecinkowych – dwa i więcej; jak i całkowitoliczbowych – cztery i więcej; dane te dotyczą typowych procesorów firmy Sun Microsystem, z około roku 1995). Następnie, procesory te mogą być umieszczone po dwa lub więcej na jednej płycie (ang. board), co powoduje, że komunikacja między nimi jest szybsza niż komunikacja z procesorami znajdującymi się na „na innej płycie". Następnie grupy płyt z procesorami są „łączone w większe grupy". Szczegóły zastosowanej topologii połączeń i jej technologicznej realizacji nie są tutaj ważne. Ważne jest to, że mamy tutaj do czynienia z co najmniej czterema poziomami hierarchii systemu wieloprocesorowego. Każdy z tych poziomów to, między innymi: inny czas dostępu do danych i inny rozmiar pamięci (jeśli weźmiemy pod uwagę: rejestry, pamięci cache L1, L2, L3, pamięć główną/pamięci lokalne). Wszystkie te aspekty sprzętowe muszą być brane pod uwagę gdy rozważana jest optymalna realizacja algorytmu równoległego na takiej maszynie. W tej sytuacji modele teoretyczne, zaproponowane w latach 80tych ubiegłego wieku, dla komputerów, które odpowiadały im technologicznie (składamy się z pojedynczych procesorów, zwykle bez pamięci cache i połączonych w homogenicznej „sieci") nie dają wyników, które mogłyby być bardzo użyteczne dla konceptualizacji efektywnych programów równoległych na współczesnych komputerach wieloprocesorowych.

(v) W pracach Kandydata nie znajdziemy również niezbędnych szczegółowych rozważań dotyczących tego w jaki sposób zrealizowane zostało zrównoleglenie kodu. W wymienionej powyżej pracy znajdujemy następujące wyjaśnienie: „algorithm uses a parallel loop, which takes as arguments the start index, number of iterations". Oznacza to, że implementacja: (a) została zrealizowana w najprostszy możliwy sposób (nie jest ona więc w żaden sposób zoptymalizowana dla rozważanego sprzętu), (b) będzie ona efektywnie działać tylko dla komputerów z pamięcią współdzieloną, (c) nie jest ona skutecznie przenoszalna. Jeśli chodzi o przenoszalność, to warto zauważyć, że wystarczyło zastosować OpenMP, które to podejście może również być wykorzystane dla maszyn z pamięcią rozproszoną (gdy liczba procesorów jest niewielka). W celu zrównoleglenia programu dla komputera składającego się z wielu procesorów (z pamięcią rozproszoną) należałoby natomiast zastosować MPI (lub rozważyć zastosowanie kombinacji OpenMP i MPI, które to podejście często jest stosowane dla poprawienia efektywności implementacji algorytmu równoległego na hierarchicznych komputerach równoległych).

(vi) W pracach Kandydata znajdujemy również pewne aspekty „metodologiczne", które są nie w pełni wyjaśnione i/lub dyskusyjne. Pierwszy z nich dotyczy porównania wyników uzyskanych przez Kandydata z innymi wynikami. Na przykład w pracy: „Modification of Parallelization of Modified Merge Sort Algorithm" na Rysunku 12, znajdujemy eksperymentalne porównanie efektywności kilku algorytmów. To co jest bardzo dziwne i niezrozumiałe to dlaczego porównana zostaje wielordzeniowa realizacja proponowanego algorytmu z jednoprocesorowymi realizacjami algorytmów HeapSort i QuickSort. Takie porównanie jest nieuprawnione metodologicznie, albowiem nie mówi ono nic o efektywności wielordzeniowej proponowanego podejścia w porównaniu z taką samą efektywnością innych algorytmów. Z podobnym problemem mamy do czynienia również w innych pracach Kandydata.

Przy okazji warto zwrócić uwagę na dziwny fakt: na Rysunku 12 mówi się o ośmiu rdzeniach, natomiast na Rysunkach 10 i 11 mówi się o ośmiu procesorach. Podobna nieprecyzyjność opisu występuje również w innych pracach Kandydata.

W kontekście rozważanej tutaj pracy warto zwrócić również uwagę na jeszcze jedną niejasność. Analizując wyniki zawarte w Tabeli 1 łatwo jest policzyć, że przyspieszenie uzyskane dla ośmiu procesorów/rdzeni wynosi ~3.87. Oznacza to, że efektywność (rozumiana zgodnie ze standardową definicją, jako iloraz uzyskanego przyspieszenia i liczby rdzeni/procesorów, wyrażana w procentach) wynosi ~48%. Nie jest więc jasnym w jakim sensie, na Rysunku 10, efektywność wynosi 70%. To prowadzi do jeszcze jednej refleksji. Z niezrozumiałych powodów Kandydat nie posługuje się standardowymi miarami sukcesu zrównoleglenia algorytmu, jakimi są przyspieszenie (ang. speed-up) i efektywność (rozumiana jak powyżej). W istotny sposób zmniejsza to zrozumiałość prac dla specjalistów z dziedziny programowania równoległego.

Podsumowując, ***pięć (z siedmiu) prac stanowiących trzon cyklu i dotyczących równoległego sortowania przez scalanie, nie stanowi znaczącego wkładu w dziedzinę.***

2.2.5 Pozostałe dwie publikacje opublikowane w międzynarodowych czasopismach

W związku z tym, że pozostałe dwie publikacje dotyczą programowania równoległego, i były opublikowane w czasopismach międzynarodowych, zostaną one omówione osobno. Rozpocznijmy analizę od publikacji: „*The analysis of energy performance in use parallel merge sort algorithm*". Praca ta została opublikowana w czasopiśmie wydawanym przez Kaunas Institute of Technology. Zawiera ona wyniki dotyczące zużycia energii w trakcie realizacji opracowanych algorytmów. Nie jest tutaj jasnym czy wyniki te pochodzą z eksperymentalnego pomiaru zużycia energii czy też z zastosowania danych dotyczących zużycia energii pochodzących z Tabeli 1, do rzeczywistego czasu sortowania. Jeżeli to nie jest rzeczywisty pomiar zużycia energii, to przedstawione wyniki nie mają większego sensu. Jak można przeczytać, na przykład, w pracy „Power-Performance Adaptation in Intel Core i7", V. Spiliopoulos, G. Keramidas, S. Kaxiras, K. Efstathiou: „i7 supports also various idle states, called C-states, in which it is possible to completely deactivate the clock and *cut-off the power supply for a combination of cores* to reduce static and dynamic power consumption" (podkreślenie MP). Oznacza to, że możliwym (i prawdopodobnym) jest, że jeżeli kod wykonuje się na jednym rdzeniu, to pozostałe rdzenie nie będą pobierały energii (energia nie będzie przez nie pobierana). W tej sytuacji wykonanie kodu na dwu rdzeniach, w czasie T/2, wcale nie musi oznaczać mniejszego wykorzystania energii niż wykonanie go na jednym rdzeniu w czasie T (jeśli pozostałe rdzenie są „uśpione"). Ponieważ nie zostało precyzyjnie opisane w jaki sposób dokonano pomiaru zużytej energii, niejasną jest więc rzeczywista wartość naukowa zaprezentowanych wyników.

Jako ostatnią rozważmy publikację „Parallel Fast Sort Algorithm for Secure Multiparty Computation". Praca ta wydaje się dotyczyć zastosowania sortowania w obszarze bezpieczeństwa. Do jakiegoś stopnia tak jest. Jednakże przedstawiona część eksperymentalna nie wychodzi poza „te same" eksperymenty, które były prezentowane w pozostałych tekstach Kandydata – czyli testowanie efektywności szybkiego algorytmu sortowania (w przypadku omawianej pracy na, wspomnianym powyżej, procesorze AMD Opteron 8356 8p). Nie ma więc praktycznego dowodu na to, że opisywany w tekście algorytm obliczeń bezpiecznych został rzeczywiście zaimplementowany.

2.3 Podsumowanie analizy jednolitego cyklu publikacji

Pozwolę sobie teraz podsumować najważniejsze wnioski wynikające z przeprowadzonej analizy jednolitego cyklu publikacji, przedstawionego do oceny przez Kandydata.

(A) Z przedstawionych do oceny 14 publikacji tylko połowa dotyczy metod równoległego sortowania. Pozostałe nie mają z tym zagadnieniem nic wspólnego.

(B) Oceniane prace nie dotyczą dużych danych, w powszechnie przyjętym rozumieniu tego terminu.

(C) Oceniane prace nie wnoszą wkładu w rzowój wiedzy o bazach NoSQL.

(D) Żadna z prac nie zawiera eksperymentów dotyczących zastosowania metod sortowania jako części „innego procesu"; wszystkie prace zawierają eksperymenty, które dotyczą tylko i wyłącznie sortowania. Oznacza to, że potencjalne zastosowania stanowią co najwyżej pewien punkt wyjścia dla rozważań, które dotyczą tylko i wyłącznie sortowania.

(E) Prace dotyczą wyłącznie sortowania wykonywanego na jednoprocesorowych odpowiednikach komputerów równoległych z pamięcią współdzieloną (równoległość na poziomie wielu rdzeni).

(F) Metoda realizacji / implementacji równoległości polega głównie na zrównolegleniu pętli, co jest najprostszym podejściem do uzyskania równoległości i oznacza, że nie da się tego podejścia efektywnie zastosować na komputerach z pamięcią rozproszoną.

(G) Eksperymenty przeprowadzone zostały dla nie więcej niż 8 procesorów/rdzeni.

(H) Przeprowadzone badania, w szczególności w części eksperymentalnej, zawierają wiele elementów dyskusyjnych/niejasnych, które istotnie obniżają ich wartość.

(I) Niektóre przedstawione wyniki powtarzają rozważania, które dużo wcześniej zostały już zaproponowane i przeanalizowane przez innych badaczy.

Tak więc należy stwierdzić, że ***wyniki badań zawarte w jednolitym cyklu publikacji stanowią marginalny wkład w rozwój informatyki technicznej w obszarze programowania równoległego.*** Ponadto, *prace te nie stanowią znaczącego wkładu w żadnym innym obszarze nauki.*

3. Ocena aktywności naukowej

Przejdźmy teraz do oceny innej działalność naukowej Kandydata.

3.1 Dane naukometryczne

Zgodnie z dostarczoną przez kandydata dokumentacją (Punkty I.2 i I.4), jest on autorem 14+15=29 publikacji. Zgodnie z punktem IV.4, zgromadził on 352 punkty zgodnie ze starym i 140 punktów zgodnie z nowym wykazem czasopism punktowanych. W tym kontekście, jeśli chodzi o wskaźniki naukometryczne, to: (a) Impact Factor wynosi 8,689, (b) w Web of Science zanotowanych jest 79 cytowań (bez autocytowań) a H index wynosi 7, (c) w Scopus odnotowanych jest 150 cytowań (najprawdopodobniej z autocytowaniami) i H index wynoszący 7, i (d) w Google Scholar 245 cytowań z autocytowaniami (na dzień 18.03.2019) i H index 10. Biorąc pod uwagę następującą rekomendację ze strony Rady Doskonałości Naukowej (https://www.rdn.gov.pl/postepowanie-habilitacyjne.wymagania-dokumentacyjne-wnioskow-w-sprawie-nadania-stopnia-doktora-habilitowanego.html):

„Jednocześnie Rada Doskonałości informuje, że podawanie danych naukometrycznych – w opinii Rady Doskonałości Naukowej – jest wskazane i zalecane, wynika to także ze stosowanej powszechnie praktyki przez samych kandydatów ubiegających się o awans naukowy. Należy jednak podkreślić, że podane we wnioskach o wszczęcie postępowania awansowego dane naukometryczne nie mogą stanowić kryterium oceny dorobku naukowego Kandydata dla podmiotów doktoryzujących, habilitujących oraz samej Rady Doskonałości Naukowej, organów prowadzących postępowania w sprawie nadania stopnia lub tytułu. Zadaniem tych organów jest przede wszystkim ***ocena ekspercka dorobku naukowego Kandydata ubiegającego się o awans naukowy, zaś decyzja o nadaniu stopnia lub tytułu nie powinna być uzależniona od podania tych danych.***" (podkreślenie MP)

chciałbym zwrócić uwagę tylko na jeden fakt, który odnosi się bezpośrednio do dorobku naukowego Kandydata. Sprawdziłem (w Google Scholar) cytowania tych prac Kandydata, które są częścią, ocenianego, jednolitego cyklu publikacji. Jak się łatwo przekonać, większość cytowań tych prac to autocytowania, lub cytowania przez współautorów. Oznacza to, że prace Kandydata, pomimo ich publikacji w liczących się konferencjach międzynarodowych i czasopismach nie wzbudziły

zainteresowania w środowisku naukowym. Ten fakt również wspiera powyższą tezę o ich minimalnym wkładzie prac Kandydata w rozwój dyscypliny.

3.2 Inne aktywności

Zgodnie z dostarczona dokumentacją, (a) Kandydat zaprezentował 11 referatów na konferencjach. Ponadto (b) był on członkiem komitetu naukowego jednej konferencji (w 2017 roku, konferencja IVUS). (c) Realizował on, lub był częścią zespołu realizującego, 3 granty, z których jeden to grant habilitacyjny. (d) Kandydat był ekspertem PARP. Ponadto (e) jest on autorem co najmniej 50 recenzji (zweryfikowanych w serwisie Publon). I na koniec, (f) jest on twórcą i przewodniczącym jury ogólnopolskiego konkursu wiedzy matematyczno-informatycznej Algorytmion. ***Aktywności te, zebrane razem, są zdecydowanie poniżej oczekiwań dla osoby, która kandyduje do habilitacji***. Warto tutaj również wspomnieć o braku funkcji promotora pomocniczego doktoratu.

Jedynym pozytywnym akcentem, jeśli chodzi o aktywność naukową Kandydata, jest pobyt na dwu 5-miesięcznych stażach w University of Catania (w latach 2018 i 2019).

4. Podsumowanie

Podsumowując niniejszą recenzję, ***działalność naukowa Kandydata w żaden sposób nie spełnia*** wymagań „Ustawy o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki" w zakresie nadania stopnia doktora habilitowanego.