dr inż. Krzysztof Różanowski
Warszawska Wyższa Szkoła Informatyki,
krozan@wwsi.edu.pl

mgr inż. Mariusz Krej
Wojskowy Instytut Medycyny Lotniczej
mkrej@wiml.waw.pl

Projekt systemu Grid Computing klasy HPC pod kątem rozwiązania zdefiniowanych zagadnień optymalizacyjnych

The project of the Grid Computing system of HPC class in terms of solving defined optimization problems

Abstrakt

W artykule przedstawiono koncepcję projektu modelowania dedykowanej platformy programistycznej do zrównoleglonego rozwiązywania zagadnień optymalizacyjnych w środowisku .NET. Jest to równoważne z udostępnieniem techniki tworzenia oprogramowania rozproszonego w wielu językach programowania, między innymi w C#, Visual Basic, ale również Fortran, Pascal, Java, Ada. Ważnym aspektem projektu jest możliwość zestawiania komputerów biurowych w potężne, wirtualne, rozproszone maszyny obliczeniowe bez zmieniania podstawowego ich przeznaczenia oraz ograniczania zakresu zastosowań.

Słowa kluczowe: obliczenia rozproszone, C#, .NET, klaster, optymalizacja, obliczenia równoległe

Abstract

The paper presents the concept of modelling of development platform dedicated to parallelized solving of optimization problems in the Microsoft .NET environment. This concept is an equivalent to the provision of distributed software development techniques in many programming languages, including C#, Visual Basic, as well as Fortran, Pascal, Java, Ada. An important aspect of the project is its ability to compile the desktop computer into a powerful virtual, distributed computing machine without changing their basic purpose and limiting their range of application.

Keywords: distributed computing, C#, .NET, grid, optimization, parallel computing

WPROWADZENIE

Przez ostatnie dziesięciolecia wzrost wydajności komputerów był zapewniany poprzez wzrost częstotliwości taktowania procesorów. Obecnie obserwujemy zatrzymanie tego procesu oraz przeniesienie wysiłków na zrównoleglenie wykonywania zadań poprzez zwiększanie liczby procesorów oraz ich potoków obliczeniowych. Jednak istnieje potężna grupa zagadnień, które cechuje ogromne zapotrzebowanie na moc obliczeniową. Przykład mogą stanowić problemy optymalizacyjne. Jednak istnieje możliwość podzielenia całkowitej pracy pomiędzy wiele komputerów. Dzięki temu łączny czas obliczeń skraca się do ułamka czasu, który zajęłoby to jednej maszynie. Istniejące rozwiązania to w dużej części oddzielne systemy operacyjne (np. Pelican, Kestrel), muszą więc zostać zainstalowane i skonfigurowane na wszystkich komputerach wchodzących w skład klastra. Planowany zrąb nie będzie stanowił oddzielnego systemu operacyjnego, działać będzie jako serwis i jako taki będzie mógł pracować w tle, nie ingerując w uruchomione przez użytkownika aplikacje [1].

Przedstawiona idea nie narzuca użytkownikowi sposobu użytkowania przez niego komputera. Modelowany klaster będzie serwisem działającym w tle. Będzie można uruchomić go na dowolnym komputerze bez potrzeby gruntownej konfiguracji komputera. Standardowe podejście polegające na budowie klastra do prowadzenia obliczeń równoległych na dużą skalę jest niezwykle kosztowne, proponowane

rozwiązanie stanowi znacząco tańszą alternatywę. Model przewiduje działanie zarówno w środowisku Windows jak i Linux, w skład klastra będą mogły wchodzić komputery działające na różnych systemach operacyjnych. System działać będzie również w systemach mikroprocesorowych opartych na architekturze ARM, zatem elementy klastra mogą stanowić dowolne urządzenia posiadające system operacyjny. Jedynym wymaganiem, jakie muszą spełnić komputery wchodzące w skład klastra jest możliwość instalacji na nim środowiska .NET lub jego ekwiwalentu.

AKTUALNY STAN WIEDZY

Podstawą systemu .NET Framework jest wspólne środowisko uruchomieniowe (ang. Common Language Runtime - CLR). Jest to maszyna wirtualna, która wykonuje kod zapisany w języku Common Intermediate Language (CIL). Środowisko uruchomieniowe CLR stanowi jeden z otwartych standardów Europejskiego Stowarzyszenia na rzecz Standaryzacji Systemów Informacyjnych i Komunikacyjnych (ang. European association for standardizing information and communication systems - ECMA), dzięki czemu może ono być implementowane przez wiele podmiotów. Oprócz wersji firmy Microsoft, do istniejących implementacji tej platformy należy zaliczyć także Mono oraz DotGNU. Możliwe jest uruchamianie kodu pisanego dla platformy .NET w wielu systemach operacyjnych. Z drugiej strony możliwe jest tworzenie kompilatorów do języka pośredniego wielu wysoko - poziomowych języków programowania, istnieje już wiele implementacji takich kompilatorów (w tym nawet Fortran, Pascal, Java, Ada). Wszystkie języki środowiska .NET , a także wszystkie biblioteki klas obecne w .NET Framework (ASP. NET, ADO.NET i inne) oparte są na CLR. Zastosowanie wspomnianego środowiska uruchomieniowego pozwoli na korzystanie przez użytkowników projektowanego zrębu z wielu różnych dostępnych języków programowania, a także pracy na wielu systemach operacyjnych.

Przy uruchamianiu projektowanego zrębu obliczeniowego w środowisku systemu operacyjnego Linux planuje się zastosowanie implementacji standardu ECMA CLR o nazwie Mono. Mono jest projektem otwartego oprogramowania założonym przez firmę Ximian, a obecnie prowadzonym przez firmę Nowell. Istnieją wersje Mono dla wielu innych systemów operacyjnych, takich jak: Android, BSD, iOS, Mac OS X, Windows, Solaris. Platforma będzie mogła znaleźć swoje zastosowanie do budowy złożonych algorytmów detekcji parametrów charakterystycznych sygnałów silnie zakłóconych mierzonych w warunkach dużej dynamiki. Numeryczne metody poszukiwania algorytmów detekcji parametrów zakłóconych sygnałów (np. sieci neuronowe, algorytmy ewolucyjne itd.) są bardzo wymagające obliczeniowo, a przy tym są podatne na zrównoleglenie [2].

Aplikacje obliczeń rozproszonych służą rozwiązywaniu problemów, które są zbyt skomplikowane obliczeniowo, aby wykonać je na jednym komputerze w akceptowalnym czasie. Obliczenia rozproszone są często stosowanie w kryptografii, również branża filmowa używa rozproszonych klastrów obliczeniowych dla produkcji filmów animowanych komputerowo.

Klaster komputerowy jest grupą serwerów i innych zasobów (zwanych węzłami), które działają jak pojedynczy system zapewniając jednocześnie wysoką dostępność usług (ang. high availability), a w niektórych przypadkach także równoważenie obciążenia (ang. load balancing) oraz przetwarzanie równoległe (ang. parallel processing). Klastrowanie polega na wykorzystaniu wielu komputerów, zwykle PC lub stacji roboczych (albo dedykowanych serwerów), wielu urządzeń przechowywania danych (macierze RAID itp.) oraz nadmiarowych połączeń pomiędzy nimi w celu utworzenia tego co użytkownicy widzą jako pojedynczy system o wysokiej dostępności. Komputery działające w technologii klastra mogą być wykorzystywane w celu równoważenia obciążenia, jak również zapewnienia wysokiej dostępności systemu (tj. nieprzerwanej, niczym niezakłóconej pracy systemu). W przypadku awarii któregoś komputera kierowane do niego żądania przejmują pozostałe węzły klastra. W praktyce rozwiązania klastrowe mają charakter mieszany i wykonują dla pewnych aplikacji funkcje wydajnościowe, przy jednoczesnym odgrywaniu roli niezawodnościowej lub równoważenia obciążenia. Szczególnie często taki tryb pracy klastra dotyczy serwerów WWW, pocztowych itp., z racji sposobu pracy aplikacji obsługujących tego typu serwisy. Z drugiej strony aplikacje bazodanowe, wykorzystywane w biznesie, słabo poddają się

zrównolegleniu (szczególnie, gdy żądamy transakcyjności). Zrównoleglenie operacji jest opłacalne jedynie w przypadku, gdy czas wykonywania obliczeń jest wyraźnie dłuższy od czasu transmisji danych między komputerami [3].

Tworzenie standardowego klastra stanowiącego system operacyjny to ingerencja skomplikowana, czasochłonna, powodująca ograniczenie innych zastosowań maszyn wchodzących w jego skład. Często wymagana jest ponowna instalacja systemu na komputerach systemu rozproszonego. W proponowanym zrębie obliczenia wykonywane będą poprzez usługę działającą w tle. Instalacja zrębu obliczeniowego na maszynie klienckiej sprowadzała się będzie do uruchomienia instalatora. Prawdopodobnie procedura ta nie wymagała będzie nawet ponownego uruchomienia komputera. System klastrowy jest odporny na awarie poszczególnych komputerów, ponieważ jest tak zbudowany, że wykluczenie jednego elementu nie wpływa negatywnie na prace pozostałej części maszyn. Podczas awarii jednej maszyny pozostałe przejmują jej zadania. Konstrukcja taka ma jeszcze jedną zaletę, pozwala na konserwację urządzeń bez wyłączenia całego sytemu. Zaletę opracowanego klastra będzie wysoka niezawodność, gdyż architektura nie wymaga, aby w jednej chwili dostępne były wszystkie węzły. Nawet w trakcie trwania obliczeń dopuszczalne będzie wyłączanie węzłów i dynamiczne dołączanie do obliczeń nowo uruchomionych węzłów. W projektowanym systemie nie będzie ograniczenia na maksymalną liczbę hostów wchodzących w skład klastra [4].

Istniejące rozwiązania w większości nie są obecnie rozwijane, albo ich obecne działanie jest niezadowalające. W istniejących zrębach narzucone jest ograniczenie na dostępne języki programowania (najczęściej język C – nie zarządzany kod) oraz na system operacyjny (najczęściej Linux).

KONCEPCJA SYSTEMU

Poszukiwane jest praktyczne rozwiązanie oparte na oprogramowaniu, a nacisk kładzie się na teorię klastrów obliczeniowych, szczegóły wykonania przy użyciu .NET oraz technologii WCF, jak również kryteria wyboru platformy do realizacji klastra. Motywacją do budowy tego klastra jest uzyskanie nowej eksperymentalnej wiedzy na temat sposobu w jaki C#, wraz z .NET i WCF mogą być stosowane w dziedzinie obliczeń rozproszonych.

System będzie posiadał architekturę rozproszoną. System składać się będzie z komputerów połączonych za pomocą protokołu TCP/IP. Jeden z komputerów pełnić będzie funkcje managera zadań, który dystrybuuje zadania dla pozostałych komputerów wchodzących w skład klastra.

Stworzone oprogramowanie obliczeń rozproszonych umożliwi zrównoleglanie algorytmów w ramach jednego bądź wielu hostów z wykorzystaniem tego samego API. Takie rozwiązanie będzie bardzo wygodne, gdyż programista podczas implementacji algorytmu może początkowo testować algorytm na własnym komputerze, nie blokując pracy klastra, a jeśli uzna że jest on gotowy do obliczeń może uruchomić go w ramach klastra obliczeniowego.

Oprogramowanie klastra zostanie zaprojektowane z uwzględnieniem wielu cech wyróżniających. Pierwszą cechą jest dynamiczna konfiguracja geometrii. W każdej chwili, nawet podczas wykonywania zadania, będzie istnieć możliwość dodania lub usunięcia węzłów. Drugą cechą jest to, że klaster będzie również odporny na błędy w działaniu mogące się pojawiać zarówno po stronie klienta, jak i serwera. System będzie działał w oparciu o środowisko Microsoft .NET. Jest to środowisko do tworzenia aplikacji komputerowych, jak również środowisko uruchomieniowe dla kodu zarządzanego. Środowisko .NET wspiera również aplikacje sieciowe, w architekturze klient serwer. Platforma pozwala na znaczne uproszczenie procesu tworzenia aplikacji. Framework dostarcza zorientowane obiektowo środowisko programistyczne jednocześnie minimalizując konflikty w dystrybuowaniu aplikacji. Kod wykonywany na platformie .NET jest w pełni zarządzany, a za kwestie bezpieczeństwa i alokacji pamięci odpowiada Common Language Runtime (CLR), za pomocą takich narzędzi jak garbage collector czy kompilator just-in-time. Ponadto biblioteka .NET zawiera bogatą kolekcję klas różnego rodzaju.

Język C# jest pierwszym językiem z rodziny C/C++, który jest zorientowany komponentowo. Jest bezpieczny, obiektowy, a narzędzia wchodzące w skład środowiska .NET (np. Visual Studio) wspierają pro-

gramowanie przy jego użyciu. Do implementacji komunikacji pomiędzy węzłami użyta zostanie technologia Windows Communication Foundation (WCF). WCF jest to interfejs programowy (ang. API) służący do budowania aplikacji w architekturze klient - serwer. WCF wykorzystuje rozmaite popularne protokoły transportowe (HTTP, TCP, Named Pipe). Jest dostępny w ramach .NET Framework jako zbiór klas (CRL). Technologia ta pozwala aplikacji udostępniać swoje API poprzez sieć, zatem aplikacja działająca na jednym komputerze jest w stanie uzyskać dostęp do serwisu udostępnianego w innej maszynie podłączonej do sieci.

Kształt interfejsu projektowanego szablonu obliczeń równoległych będzie wzorowany na bibliotece Microsoft Parallel Extensions (PFX). Stanowi ona element Microsoft .Net Framework, który został wprowadzony wraz z wersją 4.0 tej platformy. Jest to zarządzana biblioteka zrównoleglania zadań. Jej podstawowymi elementami są Parallel LINQ (PLINQ) oraz Task Parallel Library (TPL). PLINQ wspiera zrównoleglanie oparte o dane (data parallelism), w którym zbiór zadań wykonywany jest równolegle na określonym zakresie danych, a sposób implementacji jest oparty o technikę zapytań LINQ. Task Parallel Library wspiera zrównoleglanie oparte na zadaniach (task parallelism). TPL udostępnia zrównoleglone wersje pętli Foreach o oraz For. Biblioteka wspiera użytkownika w zadaniach takich jak powoływanie, zamykanie oraz określanie liczby wątków systemu operacyjnego stosowanych do wykonania obliczeń. Biblioteka PFX służy do zrównoleglania wykonywania zadań w ramach wielu rdzeni tego samego komputera. Zastosowanie sprawdzonej bazy pojęciowej wypracowanej przez Microsoft (stosowanej w ramach Parallel Extensions) powinno zapewnić przyswojenie tworzonego szablonu obliczeń rozproszonych przez szeroką grupę odbiorców. Środowisko .NET, dzięki zawartemu w nim mechanizmowi JIT (ang. just-in-time compilation) bardzo dobrze nadaje się do tworzenia aplikacji rozproszonych. JIT to metoda wykonywania programów polegająca na kompilacji do kodu maszynowego przed wykonaniem danego fragmentu kodu. W tworzonym systemie program zostanie skompilowany w taki sposób, aby maksymalnie wykorzystać architekturę procesorów będących węzłami klastra. Dzięki temu nie istnieją ograniczenia co do architektury hostów wchodzących w skład systemu.

W projektowanym systemie zostanie zaimplementowany mechanizm load balancing, który będzie przydzielał zadania węzłom w zależności od aktualnego obciążenia oraz wydajności danego węzła. Load balancing to równoważenie obciążenia, rozpraszanie żądań użytkowników aplikacji. Istnieją różne rodzaje równoważenia obciążenia. Stosowane narzędzia i techniki zależą w dużej mierze od uruchamianych aplikacji. Możliwe są tak różne przypadki, jak na przykład:

- równoważenie obciążenia serwerów web (html),
- równoważenie obciążenia serwerów bazodanowych,
- równoważenie obciążenia analiz obliczeniowych wielkiej skali (ang. large-scale computational analysis),
- równoważenie obciążenia pojedynczej maszyny wieloprocesorowej lub wielordzeniowej,
- równoważenie obciążenia wielu maszyn z pojedynczym procesorem,
- równoważenie obciążenia wielu maszyn wieloprocesorowych/wielordzeniowych,
- równoważenie obciążenia wielu maszyn działających na wielu platformach.

W zależności od rodzaju środowiska mamy do czynienia z różnymi typami obciążenia - nie istnieje więc pojedyncze rozwiązanie odpowiednie dla wszystkich scenariuszy. Do każdego środowiska potrzebny jest specyficzny scenariusz lub zestaw scenariuszy. Dlatego przy wyborze odpowiedniego mechanizmu równoważenia należy najpierw określić wymagania i środowisko działania aplikacji, których obciążenie ma zostać rozdystrybuowane. Komputer pełniący rolę serwera będzie na bieżąco monitorował aktywność poszczególnych użytkowników klastra, zmniejszając lub zwiększając obciążenie danego węzła w zależności od aktywności. Dzięki temu obliczenia rozproszone będą mogły zostać wstrzymane, w przypadku gdy użytkownik potrzebuje dużej mocy obliczeniowej do swoich zadań. Obsługa węzła wchodzącego w skład klastra nie wymaga od użytkownika jakiejkolwiek wiedzy z dziedziny informatyki, węzeł będzie łatwy i intuicyjny w obsłudze.

Zakłada się, że w opracowywane rozwiązanie zostanie zrealizowane w 100% z wykorzystaniem języku wysokiego poziomu C#. Nie istnieje potrzeba programowania na niskim poziomie (assembler), aby zrealizować zakładany cel. Z pewnością będzie to zaletą podczas rozwoju oprogramowania. Jednak w razie potrzeby, możliwe będzie oprogramowanie wrappera do wykonania w środowisku klastra kodu z dynamicznej biblioteki języka C. Niemniej jednak środowisko .NET jest znacznie bardziej preferowane do użycia w tym przypadku ze względu na uruchamianie aplikacji z poziomu maszyny wirtualnej i mechanizm kompilacji w locie JIT. Dzięki tym mechanizmom raz skompilowany program ma szanse wykonać się szybko na różnych architekturach procesora „dostrajając się" do konkretnego modelu procesora. Należy zauważyć, że opracowywany klaster będzie wykorzystywał do działania przede wszystkim standardowe komputery typu PC pracujące pod kontrolą środowiska Windows i Linux. Każdy z tych komputerów potencjalnie będzie posiadał inną konfigurację sprzętową. Wybór środowiska Microsoft .NET do tego celu wydaje się rozwiązaniem idealnym. W przypadku użycia dynamicznej biblioteki języka C należałoby przygotować wiele wersji kompilacji tej biblioteki, aby na każdym komputerze wykorzystać możliwości zainstalowanego procesora. Środowisko Microsoft .NET zapewnia tym samym znacznie większą skalowalność i przenośność tworzonych programów. Dopuszcza się także możliwość podłączenia do tworzonego klastra profesjonalnych komputerów obliczeniowych bądź serwerowych, aby wykorzystać ich wolną moc obliczeniową.

Obecnie na rynku nie sprzedaje się już komputerów PC jedno rdzeniowych (poza tanimi komputerami typu netbook). Większość komputerów posiada 2, 4, a nawet 8 rdzeni logicznych zawartych w ramach architektury SMP (ang. Symmetric Multiprocessing). W przypadku pracy biurowej na komputerze w zdecydowanej większości czasu wykorzystywany jest jedynie 1 rdzeń procesora. Wynika to z kilku powodów. Po pierwsze, nie wszystkie aplikacje wymagają przetwarzania równoległego. Przykładowo użytkownik podczas pisania tekstu na klawiaturze nie jest w stanie wykorzystać choćby w kilku procentach 1 rdzenia współczesnego procesora. Drugim powodem jest opieszałość producentów oprogramowania w tworzeniu nowoczesnego kodu wykorzystującego wiele procesorów. Wciąż spotyka się oprogramowanie wykorzystujące jeden rdzeń procesora, chociaż nie widać przeciwwskazań do wykorzystania wielu rdzeni (np. konwertery formatu obrazu video). Można bez obaw stwierdzić, iż moc komputerów biurkowych jest niewykorzystana w bardzo znacznym procencie. Tworzony klaster ma umożliwić skorzystanie naukowcom i inżynierom z tej niewykorzystanej mocy obliczeniowej.

Przewiduje się opracowanie mechanizmu wykrywania aktywności użytkownika i dynamiczne sterowanie obciążeniem, aby przeprowadzane obliczenia nie przeszkadzały w pracy na komputerze. Konieczne będzie zaprojektowanie systemu zabezpieczeń (uwierzytelnienie, autoryzacja, kontrola wykonywanego algorytmu), aby użytkownik węzła nie musiał obawiać się o bezpieczeństwo danych przechowywanych na komputerze.

Opracowany klaster będzie mógł być wykorzystany do przeprowadzenia obliczeń polegających na poszukiwaniu rozwiązań problemów np. z wykorzystaniem algorytmów genetycznych, jak również może wspierać aplikację w swoim działaniu w trybie bieżącym (ang. realtime) poprzez oddelegowywanie złożonych obliczeń do innych hostów (np. silna kompresja materiału video w locie). Aby było to możliwe opracowane zostaną biblioteki dynamiczne DLL z pełnym interfejsem API dla programistów do obsługi i konfiguracji węzłów klastra. Zapewniony zostanie mechanizm transportu plików z węzła centralnego do pozostałych węzłów, aby możliwe było przetransportowanie danych wejściowych dla algorytmów, bądź odbioru przetworzonych plików.

METODYKA BADAŃ

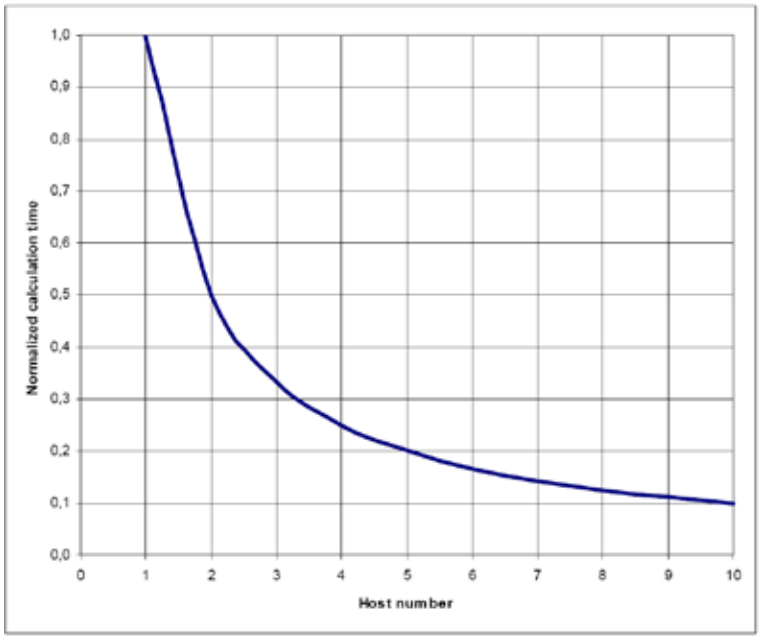
Model zostanie wytworzony przy zastosowaniu najnowszych technik i metod projektowania, wytwarzania oraz testowania oprogramowania. W trakcie analizy modelu zostaną użyte nowoczesne i sprawdzone narzędzia do tworzenia i testowania systemów informatycznych. W ramach analizy,

w celu zapewnienia niezawodności systemu, stworzony zostanie zestaw oprogramowania automatyzującego proces testowania. Zakłada się, że powinien on pozwalać na zbadanie prawidłowości działania wszystkich komponentów tworzonego systemu. Główny zrąb oprogramowania zostanie zamodelowany w języku C# zgodnie z paradygmatem programowania obiektowego. W procesie projektowania zastosowane zostaną wzorce projektowe.

Do zalet tworzenia aplikacji w środowisku wirtualnym, jakim jest platforma .NET Framework należy zaliczyć automatyczne zarządzenie pamięcią, automatyczna kontrola „życia" obiektów (Garbage Collector), odporność na krytyczne błędy aplikacji, możliwość pisania w wielu językach programowania w ramach jednego projektu, niezależność od platformy sprzętowej, wysokie poziom bezpieczeństwa (np. poprzez zastosowanie sandboxing'u) oraz bogate możliwości analizy działania aplikacji.

Badania z użyciem zamodelowanego klastra będą polegać na wykonywaniu złożonych obliczeń zarówno na pojedynczym komputerze, jak i na wielu komputerach wchodzących w skład klastra oraz porównywaniu czasu trwania obliczeń. Celem badań jest znalezienie korelacji między ilością węzłów wchodzących w skład klastra, a szybkością wykonywania obliczeń. Testowana będzie również odporność powstałego systemu na błędy pojawiające się w trakcie jego działania.

W ramach badań opracowywanego systemu zostanie wyznaczony rzeczywisty przebieg zależności przyspieszenia wykonywania obliczeń od liczby komputerów wchodzących w skład klastra dla poszczególnych możliwych konfiguracji zrębu oraz specyficznych rodzajów wykonywanych zadań. Korzyści z przeprowadzania obliczeń w systemie rozproszonym przedstawia poniższy wykres.



Rys. 1 Wykres teoretycznego przebiegu wzrostu wydajności w systemie rozproszonym.

Na wykresie przedstawiono idealny, teoretyczny przebieg wzrostu wydajności. W praktyce przyśpieszenie obliczeń jest mniejsze i w dużym stopniu zależy od samego algorytmu podlegającego zwielokrotnieniu oraz umiejętności programisty. Należy pamiętać, że znaczna część algorytmu może nie być możliwa do zwielokrotnienia. Przykładowo, nie da się w prosty sposób zwielokrotnić wykonywania pętli, w której elementy iteracji zależą od iteracji poprzedniej. Poza tym czasy transmisji danych między węzłami także wpływają na ostateczny bilans czasu wykonania algorytmu na wielu hostach. Nie mniej jednak znaczna część algorytmów podlega zwielokrotnieniu i zauważa się trend do modyfikacji algorytmów do wersji mogących wykonywać się równolegle (np. algorytmy transformaty Fouriera). Górny limit przyspieszenia jest określony przez prawo Amdahl'a znanym również jako wywód Amdahla [5]. Prawo to używane jest do znajdywania maksymalnego zwiększenia wydajności obliczeń całego syste-

mu jeżeli tylko cześć systemu została usprawniona. Prawo to często wykorzystuje się do przewidywania teoretycznego wzrostu szybkości obliczeń w przypadku zwielokrotnienia fragmentów algorytmu na wiele procesorów (komputerów).

Należy pamiętać, że nawet ramach jednego komputera mimo obecności przykładowo 4 rdzeni ALU w jednym procesorze nigdy nie otrzymamy wzrostu wydajności równego 4 ze względu chociażby na ograniczenia magistrali pamięci RAM, która nie pozwala w obecnych komputerach na odpowiednio szybką i niezależną wymianę danych między pamięcią RAM a procesorami, aby te mogły pracować zupełnie równolegle i niezależnie. Skutkiem tego jest blokowanie pracy rdzenia procesora w oczekiwaniu na dostęp pamięci. Poza tym w komputerach występuje wiele sekwencyjnych urządzeń wejścia-wyjścia (np. dysk twardy), które mogą powodować przestoje w wykonywaniu algorytmu. Odpowiednie algorytmy i buforowania pomagają równoważyć opóźnienia i narzuty związane z synchronizacją wyżej omówioną.

Podstawowym wskaźnikiem definiującym wzrost wydajności przeprowadzania obliczeń w systemie obliczeń rozproszonych może być stosunek czasu wykonania algorytmów testowych na jednym rdzeniu logicznym procesora do czasu wykonania algorytmu na wielu rdzeniach logicznych przy wykorzystaniu wielu hostów obliczeniowych. Wartość wynikowa będzie wyrażała wzrost wydajności (przyspieszenia obliczeń). Eksperyment pomiaru będzie wykonywany wielokrotnie (nawet setki razy, w sposób automatyczny) dla wielu algorytmów, aby uzyskać rzeczywisty uśredniony wzrost wydajności. Powtórzenie eksperymentu będzie wymagane aby wyeliminować przypadki losowe zaburzające pomiary (np. chwilowe obciążenie sieci Ethernet, uruchomienie na węźle narzędzi konserwacji systemu takich jak defragmentator dysków itp.). Opracowany system zostanie zweryfikowany pod kątem zgodności z prawem Amdahla, kiedy to tylko część algorytmu podlega zwielokrotnieniu. Zbadana także zostanie wydajność systemu przy zwielokrotnieniu drobnoziarnistym i gruboziarnistym.

Poza testami wydajnościowymi zrealizowane zostaną testy niezawodnościowe systemu. Zbadany będzie wpływ systemu na zasymulowane uszkodzenia, takie jak:

- niedostępność jednego z węzłów,
- odłączenie węzła w trakcie trwania obliczeń,
- próby przeciążania sieci w trakcie trwania obliczeń celem zwiększenia opóźnień przesyłanych danych.

Architektura systemu zapewniać będzie odporność na błędy transmisji danych i niedostępność węzłów, stąd spodziewany jest pozytywny wynik zachowania opracowanego systemu obliczeniowego na losowe uszkodzenia. Zaawansowany zarządca zadań będzie nadzorował zlecane zadania do wykonania i w przypadku niepowodzenia, zadania kierowane będą do innych węzłów.

PODSUMOWANIE

W omawianym rozwiązaniu będzie dostępna możliwość zrównoleglenia drobnoziarnistego oraz gruboziarnistego. Przy czym należy pamiętać o narzucie przy przesyłaniu danych przez dostępne interfejsy sieciowe. W takich przypadkach zrównoleglenie drobnoziarniste przeważnie jest efektywniejsze, gdyż czas transmisji potrzebnych danych jest stosunkowo pomijalny do czasu przetwarzania tych danych. W przypadku zrównoleglenia drobnoziarnistego czasy transmisji mogą mieć istotne znaczenie. W pewnych szczególnych przypadkach zrównoleglenie może w ogóle się nie opłacać (nie opłaca się przykładowo oddelegować do wykonania do innego hosta przykładowego algorytmu dodającego dwie liczby 32 bitowe, gdyż narzut związany z przygotowaniem i wysłaniem danych oraz odebraniem wyniku będzie wielokrotnie większy niż wykonanie tych obliczeń przez sam procesor zlecający). Istniejące narzędzia zrównoleglenia takie jak obliczenia z zastosowaniem kart graficznych lub narzędzia oparte o wyrażenia typu lambda rozwijane do drzew wyrażeń (ang. Expression Trees) preferują zrównoleglenie drobnoziar-

niste. Jedną z ważnych zalet proponowanego sposobu realizacji zrębu obliczeniowego jest możliwość zrównoleglenia wielkich i skomplikowanych gałęzi kodu.

Zakłada się, iż opracowane rozwiązanie zostanie opublikowane na zasadach wolnego oprogramowania. Umożliwi to współtworzenie projektu przez ochotników w szerokim zakresie. Należy tu także podkreślić zarówno możliwość testowania oprogramowania przez niezależnych użytkowników, ale także weryfikacji, poddania dyskusji oraz krytyce proponowanego podejścia. W przypadku pozytywnego uruchomienia zrębu obliczeń rozproszonych i dobrego przyjęcia stworzonego oprogramowania w gronie społeczności programistów Microsoft .NET możliwe będzie poszerzanie klastra o komputery z innych instytucji zainteresowanych współpracą, a nawet wynajem mocy obliczeniowej komputerów firmom trzecim. Obliczenia takie mogły by przykładowo wykonywać się w nocy. Oczywiście jest to uwarunkowane odpowiednim zabezpieczeniem komputerów przeciw ujawnieniu prywatnych danych.

BIBLIOGRAFIA

[1] W. Lin, Ch. Guo, D. Qi, Y. Chen, Z. Zhili, „Implementations of Grid-Based Distributed Parallel Computing," Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on , vol.1, no., pp.312-317, 20-24 June 2006.

[2] M. Ku, D. Min, E. Choi, „SCAREX: A framework for scalable, reliable, and extendable cluster computing," Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on , vol., no., pp.966-972, Nov. 30 2010-Dec. 2 2010.

[3] A. Simon, M. Adjouadi, M. Ayala, „A .NET solution for distributed computing applications," Potentials, IEEE , vol.25, no.2, pp. 24- 28, March-April 2006.

[4] G, Xu, F. Lu, H. Yu, Z. Xu, „A Distributed Parallel Computing Environment for Bioinformatics Problems," Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on , vol., no., pp.593-599, 16-18 Aug. 2007.

[5] LJ. Zhang, Q. Zhou, „CCOA: Cloud Computing Open Architecture," Web Services, 2009. ICWS 2009. IEEE International Conference on , vol., no., pp.607-616, 6-10 July 2009.