HPC na konkurencyjnych zasobach w chmurze

Obliczenia jako narzędzie weszły do głównego nurtu. Naukowcy mogą teraz łatwo wynajmować czas w dużych klastrach komercyjnych, które można rozbudowywać i zmniejszać na żądanie w czasie rzeczywistym. Jednak obecna wydajność komercyjnego przetwarzania w chmurze nie spełnia wymagań systemów zaprojektowanych specjalnie do zastosowań naukowych. Potrzeby w zakresie obliczeń naukowych różnią się znacznie od potrzeb aplikacji internetowych, na których skupiali się dostawcy usług w chmurze. W tym rozdziale zademonstrujemy poprzez ocenę empiryczną wydajność obliczeniową wysokowydajnych aplikacji numerycznych w komercyjnym środowisku chmury, gdy zasoby są współdzielone w warunkach dużej rywalizacji. Wykorzystując benchmark Linpack jako studium przypadku, pokazujemy, że wykorzystanie pamięci podręcznej staje się wysoce nieprzewidywalne i podobnie wpływa na czas obliczeń. W przypadku niektórych problemów nie tylko bardziej wydajne jest niepełne wykorzystanie zasobów, ale rozwiązanie można znaleźć szybciej w czasie rzeczywistym (w czasie rzeczywistym). Pokazujemy również, że najmniejsza, najtańsza (64-bitowa) instancja w badanym środowisku ma najlepszy stosunek ceny do wydajności. W świetle dużej kontrowersji, której jesteśmy świadkami, uważamy, że należy wprowadzić alternatywne definicje wydajności dla komercyjnych środowisk chmurowych tam, gdzie nie istnieją silne gwarancje wydajności. Pojęcia takie jak średnia, oczekiwana wydajność i czas wykonania, oczekiwany koszt do ukończenia oraz miary wariancji — tradycyjnie ignorowane w kontekście obliczeń o wysokiej wydajności - powinny teraz uzupełniać lub nawet zastępować standardowe definicje wydajności.

Wstęp

Model przetwarzania w chmurze kładzie nacisk na możliwość skalowania zasobów obliczeniowych na żądanie. Korzyści dla użytkowników są liczne. W przeciwieństwie do konwencjonalnych systemów klastrowych, nie ma znaczących nakładów finansowych ani czasowych z góry na infrastrukturę ludzi i bieżące wydatki są uproszczone. Gdy zasoby nie są używane, całkowity koszt może być bliski zeru. Zamiast alokować zasoby według średniego lub szczytowego obciążenia, użytkownik chmury może pokryć koszty wprost proporcjonalne do bieżących potrzeb. Osoby fizyczne mogą szybko tworzyć i skalować niestandardowy klaster obliczeniowy, płacąc tylko za sporadyczne użytkowanie. Jednak są też pewne wady. Koszty można podzielić na różne kategorie rozliczane osobno: na przykład sieć, pamięć masowa i użycie procesora. Ten model może być złożony, gdy próbuje się minimalizować koszty. Co więcej, czas konfiguracji zasobów obliczeniowych jest obecnie dość długi (rzędu minut w przypadku Amazon), a ziarnistość rozliczania zasobów procesora jest zgrubna: co godzinę. Te dwa czynniki sugerują, że zasoby powinny być bardzo ostrożnie skalowane w obecnych chmurach, zmniejszając niektóre korzyści ze skalowania na żądanie. Wreszcie, w wielu środowiskach chmurowych zasoby fizyczne są współdzielone między węzły wirtualne należące do różnych użytkowników, co może negatywnie wpływać na wydajność. Możliwość przydzielania węzłów na żądanie w obecnych komercyjnych środowiskach chmurowych oznacza nowy problem w obliczeniach naukowych: rywalizację. Dobre klastry obliczeniowe o wysokiej wydajności równoważą wykorzystanie procesora, pamięci i sieci przez aplikacje w celu utrzymania wydajności przy jednoczesnym zwiększaniu wykorzystania zasobów, ale zakładają wyłączny dostęp do tych zasobów przez aplikację. Na przykład superkomputer RoadRunner w Los Alamos National Laboratory może rozwiązać system liniowy wykorzystujący 100 tys. rdzeni przy zachowaniu 90% wydajności wykorzystania procesora. Ten wyczyn wymaga starannego zrównoważenia potrzeb aplikacji między procesorem, pamięcią i siecią. Dzięki wyłącznemu dostępowi programiści aplikacji i bibliotek obliczeniowych mogą osiągnąć takie zrównoważone przetwarzanie poprzez zmniejszenie szumu związanego z wykorzystaniem zasobów, który uniemożliwia wydajną synchronizację. Na przykład jeden węzeł z dużego systemu może uruchamiać dodatkowe oprogramowanie do monitorowania, które powoduje, że osiąga koniec

obliczeń wolniej niż inne węzły. Jeśli wszystkie węzły muszą zostać zsynchronizowane na koniec obliczeń, wszystkie inne węzły w systemie muszą czekać na zakończenie najwolniejszego węzła. Niewielkie ilości szumu mogą znacząco wpłynąć na ogólną wydajność ściśle powiązanej aplikacji. Chociaż środowiska chmurowe zazwyczaj zapewniają niewielki stopień gwarancji dostępności zasobów, nie zapewniają pełnego faksymile nieudostępnionego węzła wirtualnego ani silnych gwarancji wydajności. Rywalizacja jest widoczna w prawie wszystkich zasobach: przepustowości pamięci, przestrzeni pamięci podręcznej ostatniego poziomu, przepustowości sieci i opóźnieniu, a nawet czasu procesora. Oprócz wprowadzenia omówionego powyżej rodzaju szumu, taka rywalizacja może poważnie ograniczyć wykorzystanie zasobów współdzielonych między środowiskami wirtualnymi. Na przykład rywalizacja o pamięć może zwiększyć czas obliczeń prostego mnożenia macierzy o rząd wielkości.

Aby skutecznie uruchamiać aplikacje naukowe, przetwarzanie w chmurze musi zapewniać aplikacjom naukowym zasoby porównywalne z obecnymi dobrze zrównoważonymi, ekskluzywnymi systemami obliczeniowymi o wysokiej wydajności (HPC). Izolacja między różnymi maszynami wirtualnymi w chmurze powinna zapewnić przewidywalną wydajność deweloperom aplikacji i bibliotek wymagających dużej mocy obliczeniowej. Jednak nasza ocena chmury Amazon EC2 – obecnie największego komercyjnego środowiska chmury – pokazuje, że nadal potrzebne są znaczne prace, aby wyizolować maszyny wirtualne w chmurze lub dostosować biblioteki HPC do dynamicznego, kontrowersyjnego środowiska chmury. System chmurowy Amazon EC2 został zbudowany z myślą o obciążeniach usług internetowych (tj. bez niezwykle szybkich połączeń) i nie może być korzystnie w porównaniu z celowo zbudowanymi, mocno zaprojektowanymi superkomputerami HPC dla wielu ściśle powiązanych obliczeń naukowych, takich jak gęsta algebra liniowa. Chociaż nie oczekujemy porównywalnej wydajności, dynamiczna skalowalność środowisk daje pewne nadzieje dla mniejszych obciążeń. Aby określić osiągalną wydajność obecnych systemów chmurowych dla HPC, rozważamy wykonanie gęstej algebry liniowej, która zapewnia korzystny stosunek obliczeń do komunikacji: operacje O(n3) na danych O(n2). Gęste algorytmy algebry liniowej mogą zatem nakładać się na powolną komunikację danych z szybkimi obliczeniami na znacznie większej ilości danych. Systemy chmurowe będą generalnie ograniczone przez powolną komunikację bardziej niż wyspecjalizowane systemy HPC, biorąc pod uwagę ich stosunkowo tanie, powolne połączenia. Martwimy się przede wszystkim wydajnością wspólnych systemów chmurowych, ponieważ skalują się one i pokazują, że skutki rywalizacji mogą znacznie przeważyć (przynajmniej w bieżących ofertach) niezbilansowanie spowodowane przez niedostarczone połączenie międzysieciowe. Skupiamy się zatem na kosztach procesora, ignorując możliwe powiązane (i znacznie mniejsze na obliczenia) koszty pamięci masowej i sieci. Ten rozdział empirycznie bada wydajność obliczeniową w środowisku chmury, gdy zasoby są współdzielone w warunkach dużej rywalizacji. Rozważamy wydajność pojedynczego węzła w gęstych operacjach algebry liniowej, podczas gdy węzły są współdzielone z nieznanymi innymi aplikacjami na Amazon EC2. Ponadto analizujemy, jak zmienia się wydajność w klastrach zawierających do 64 rdzeni obliczeniowych. Nasze wyniki pokazują, że wydajność pojedynczych węzłów dostępnych w EC2 może być tak dobra, jak węzły znalezione w obecnych systemach HPC, ale średnia wydajność jest znacznie gorsza i wykazuje dużą zmienność. W rzeczywistości gęste algorytmy algebry liniowej nie wydają się dobrze skalować w systemach chmurowych o wysokim poziomie rywalizacji. Rywalizacja wypacza równowagę między dostępnością zasobów procesora i pamięci węzła a przepustowością sieci, aby uniemożliwić efektywne wykorzystanie zasobów. Wykorzystanie pamięci podręcznej staje się wysoce nieprzewidywalne i podobnie wpływa na czas obliczeń. Pokazujemy, że w przypadku niektórych problemów nie tylko niepełne wykorzystanie zasobów procesora jest bardziej wydajne, ale rozwiązanie można nawet uzyskać szybciej w czasie rzeczywistym (czas na ścianach). W świetle dużej kontrowersji, której jesteśmy świadkami, uważamy, że należy wprowadzić alternatywne definicje wydajności dla

środowisk chmurowych tam, gdzie nie istnieją silne gwarancje wydajności. Pojęcia takie jak średnia oczekiwana wydajność i czas wykonania, oczekiwany koszt do ukończenia oraz miary wariancji — tradycyjnie ignorowane w kontekście HPC - powinny teraz uzupełniać lub nawet zastępować standardowe definicje wydajności. Oprócz standardowych wskaźników, takich jak GFLOP i wydajność wykorzystywanych w HPC, wprowadzamy $/GFLOP (dolary na miliard operacji zmiennoprzecinkowych) w celu dogłębnej analizy zalet i wad chmur. Metryka $/GFLOP pozwala użytkownikom oszacować proste koszty — obecnie ograniczone do wykorzystania procesora - dla różnych aplikacji pod względem wydajności obliczeniowej.

Powiązana praca

Przetwarzanie w chmurze zostało zaproponowane jako usługa umożliwiająca skalowanie lub udostępnianie istniejących klastrów naukowych. Projekt Eucalyptus zapewnia narzędzie do zarządzania chmurą typu open source. Podobnie projekt Nimbus zapewnia narzędzia do zarządzania chmurą dla klastrów. Narzędzia te zazwyczaj nie zapewniają obsługi wielu dzierżawców - w przeciwieństwie do analizowanych przez nas komercyjnych środowisk chmurowych - i tym samym obserwują różne charakterystyki wydajności bliższe istniejącym klastrom naukowym z wyłącznym dostępem. W tym rozdziale zajmiemy się skutkami rywalizacji na aplikacje HPC, gdy wiele maszyn wirtualnych współużytkuje zasoby. Tikotekar i in. uruchamiać różne testy porównawcze HPC na maszynach wirtualnych, aby określić wpływ wirtualizacji na aplikacje obliczeniowe o wysokiej wydajności. Wyniki pokazują, że izolowanie wpływu wielu maszyn wirtualnych na tę samą maszynę jest trudne, ale ogólnie wirtualizacja ma niewielki wpływ na wydajność aplikacji HPC wymagających dużej mocy obliczeniowej. Podobnie Youseff i inni widzą niewielki wpływ wydajności na zachowanie hierarchii pamięci bibliotek algebry liniowej podczas wirtualizacji. Jednak to badanie nie uwzględnia rywalizacji między maszynami wirtualnymi z wieloma dzierżawcami. Wpływ wirtualizacji na wydajność sieci EC2 jest analizowany empirycznie przez Wang i Ng (2010). Zgłaszają duże niestabilności i opóźnienia sieci, zwłaszcza w małych 32-bitowych węzłach. Obserwujemy słabą wydajność w równoległym teście porównawczym HPL przy użyciu wielu instancji, ale bardziej interesują nas właściwości skalowania. Chociaż instancje nie osiągają szczytowej wydajności, równoległe zadanie może się skalować (dla dziesiątek węzłów) ze słabą wydajnością sieci, pod warunkiem, że instancje mają wystarczającą ilość zainstalowanej pamięci RAM ze względu na wysoki stosunek obliczeń do komunikacji w gęstej algebrze liniowej. Nasze eksperymenty pokazują ten efekt. Efekty współdzielenia pamięci podręcznej ostatniego poziomu w procesorach wielordzeniowych omawiają Iyer i inni. W artykule omówiono różne podejścia do poprawy gwarancji wydajności każdego rdzenia w odniesieniu do zachowania pamięci podręcznej. Korzystanie z tych technik może znacznie zwiększyć przewidywalność wydajności na platformach chmurowych. Wcześniej Edward Walker porównał węzły EC2 z obecnymi systemami HPC . Nasze wyniki tutaj są podobne do jego dla małych klastrów 4 węzłów, których użył.

Tło

Przeprowadzamy nasze eksperymenty na usłudze Amazon Elastic Compute Cloud (EC2) jako studium przypadku dla komercyjnych środowisk chmurowych. Chociaż istnieją konkurencyjne oferty chmury, które były wówczas publicznie dostępne, usługa Amazon jest największą, która zapewnia wysoce konfigurowalne maszyny wirtualne. Węzły przydzielone przez EC2 uruchamiają jądro lub system operacyjny skonfigurowany przez Amazon, ale całe oprogramowanie powyżej tego poziomu jest konfigurowane przez użytkownika. Wiele innych ofert chmurowych innych dostawców ogranicza aplikacje do określonych interfejsów API lub języków. Aby wykorzystać istniejące, wysoce zoptymalizowane biblioteki gęstej algebry liniowej, jako studium przypadku używamy Amazon. Węzły przydzielone przez EC2 nazywane są instancjami. Instancje są przydzielane z centrów danych Amazon

zgodnie z nieopublikowanymi algorytmami planowania. Przydziały są początkowo ograniczone do 20 instancji, ale to ograniczenie można znieść na żądanie. Centra danych są łączone w jednostki znane jako strefa dostępności, najmniejsza logiczna jednostka geograficzna Amazon do alokacji. Strefy te są następnie łączone w regiony, które obecnie składają się tylko ze Stanów Zjednoczonych i Europy. Po przydzieleniu każda instancja automatycznie ładuje określony przez użytkownika obraz zawierający odpowiedni system operacyjny (w naszym przypadku Linux) i oprogramowanie użytkownika (opisane poniżej). Obrazy są automatycznie ładowane przez usługi Amazon na jeden lub więcej zwirtualizowanych procesorów przy użyciu maszyny wirtualnej (VM) Xen . Każdy procesor jest sam w sobie wielordzeniowy, co daje w sumie 2–8 wirtualnych rdzeni dla zarezerwowanych przez nas instancji. Warunki świadczenia usług przez Amazon nie dają silnych gwarancji wydajności. Co najważniejsze dla tego badania, nie ograniczają one zdolności Amazon do wdrażania wielu najemców; oznacza to wspólne lokalizowanie maszyn wirtualnych od różnych klientów. Poniżej omówimy charakterystykę wydajności różnych instancji. Narzędzia napisane do publicznych interfejsów API firmy Amazon umożliwiają przydzielanie dodatkowych węzłów na żądanie, zwalnianie nieużywanych węzłów oraz tworzenie i niszczenie obrazów, które mają być ładowane na przydzielone instancje. Korzystając z tych narzędzi i opracowując własne, zbudowaliśmy obrazy przy użyciu najnowszych kompilatorów dostarczonych przez producentów procesorów AMD i Intel. Używamy HPL 2.0 z University of Tennessee, skompilowanego z GotoBLAS 1.26 z Texas Advanced Computing Center (TACC) i MPICH2 1.0.8 z Argonne National Laboratory. Korzystając z naszych narzędzi możemy automatycznie przydzielać i konfigurować klastry o zmiennej wielkości w EC2, w tym obsługę aplikacji MPI. Chociaż opracowaliśmy narzędzia do automatycznego zarządzania i konfiguracji węzłów EC2 dla naszych aplikacji, istnieją również inne publicznie dostępne narzędzia do uruchamiania aplikacji naukowych na platformach chmurowych (w tym EC2) (chmury Nimbus Science). Co więcej, w miarę dojrzewania platformy przetwarzania w chmurze spodziewamy się znacznie większego rozwoju konkretnych aplikacji, takich jak obliczenia o wysokiej wydajności w celu zmniejszenia lub wyeliminowania znacznej części początkowej krzywej uczenia się przy wdrażaniu aplikacji naukowych na platformach chmurowych. Na przykład, obrazy publiczne są już dostępne na EC2 obsługującym MPICH.

Przegląd konfiguracji Amazon EC2

Nasze studium przypadku zostało przeprowadzone przy użyciu różnych typów instancji w usłudze Amazon Elastic Compute Cloud (EC2) od listopada 2008 do stycznia 2010. Tabela opisuje istotne różnice między typami instancji: liczba rdzeni na instancję, zainstalowana pamięć, teoretyczna szczytowa wydajność , a koszt instancji na godzinę.

Instance	Processor	Cores	RAM (GB)	Peak (Gflops)	Price ($/hr)
m1.large	Intel Xeon E5430	2	7.5	21.28	$0.34
m1.large	AMD Opteron 270	2	7.5	8.00	$0.34
m1.xlarge	Intel Xeon E5430	4	15	42.56	$0.68
m1.xlarge	AMD Opteron 270	4	15	16.00	$0.68
c1.xlarge	Intel Xeon E5345	8	7	74.56	$0.68
m2.2xlarge	Intel Xeon X5550	4	34.2	42.72	$1.20
m2.4xlarge	Intel Xeon X5550	8	68.4	85.44	$2.40

Używaliśmy tylko instancji z procesorami 64-bitowymi, więc traktujemy m1.large jako najmniejszą instancję, chociaż Amazon dostarcza mniejszą 32-bitową instancję m1.small. Koszty na węzeł różnią się 7 razy od 0,34 USD za najmniejszy do 2,40 USD za węzły ze znaczną ilością zainstalowanej pamięci. Zauważamy, że koszty skalują się bardziej w przypadku zainstalowanej pamięci RAM niż w przypadku szczytowej wydajności procesora - instancja c1.xlarge jest odstająca. Szczytowa wydajność jest

obliczana na podstawie możliwości procesora. Na przykład typ wystąpienia c1.xlarge składa się z 2 czterordzeniowych procesorów Intel Xeon działających z częstotliwością 2,3 GHz i łącznej pamięci 7 GB. Każdy rdzeń jest w stanie wykonać 4 operacje zmiennoprzecinkowe na cykl zegara, co prowadzi do teoretycznej szczytowej wydajności 74,56 GFLOP na węzeł. Istnieją dodatkowe koszty przepustowości wykorzystywanej do i z sieci Amazon oraz długoterminowego przechowywania danych, ale ignorujemy te koszty w naszych obliczeniach, ponieważ są one znikome w porównaniu z kosztami samych obliczeń w naszych eksperymentach. Jeśli chodzi o wielowątkową równoległość zapewnianą przez procesory wielordzeniowe, szeroko zakrojone testy zwykle zapewniały najlepszą wydajność, gdy ustawiliśmy bibliotekę Goto BLAS tak, aby używała tylu wątków, ile jest dostępnych rdzeni na gniazdo - 4 i 2, odpowiednio dla Xeona i Opterona . Liczbę wątków użytych do uzyskania konkretnych wyników podajemy w dalszej części, przedstawiając szczytową osiągniętą wydajność. Dzięki tym ustawieniom i użyciu bibliotek i kompilatorów specyficznych dla platformy osiągnęliśmy 76% i 68% teoretycznej wydajności szczytowej (mierzonej w GFLOP/s) odpowiednio dla Xeon E5345 i Opteron 270 dla wydajności pojedynczego węzła w dużych instancjach. Uważamy zatem, że konfiguracja i wykonanie LINPACK w HPL na instancjach o wysokim procesorze i standardowych jest wystarczająco wydajna, aby można ją było wykorzystać jako przykład aplikacji intensywnie korzystających z mocy obliczeniowej do celów naszej oceny. Wszystkie typy instancji (z procesorami Intel lub AMD) uruchamiają system operacyjny RedHat Fedora Core 8 przy użyciu jądra Linux 2.6.21. Linia 2.6 jąder Linuksa obsługuje automatyczne dostrajanie rozmiarów buforów dla sieci o wysokiej wydajności, co jest domyślnie włączone. Konkretny interkonekt używany przez Amazon jest nieokreślony (Services, 2010), a wiele instancji może nawet współdzielić pojedynczą sprzętową kartę sieciową.

W związku z tym cała przepustowość może nie być dostępna dla żadnego konkretnego wystąpienia. Aby jak najlepiej zredukować liczbę przeskoków między węzłami, przeprowadzamy wszystkie eksperymenty z węzłami klastra przydzielonymi w tej samej strefie dostępności.

Przegląd HPL

Naszym celem jest określenie przydatności komercyjnych środowisk chmurowych do określonych rodzajów zastosowań naukowych. Skupiamy się na benchmarku HPL jako przykładzie ściśle powiązanych, wysoce równoległych zastosowań naukowych. HPL oblicza rozwiązanie losowego gęstego układu równań liniowych za pomocą faktoryzacji LU z częściowymi rozwiązaniami obrotowymi i trójkątnymi. Algorytm ten wymaga operacji zmiennoprzecinkowych O(n3) na danych O(n2); oznacza to, że HPL wymaga dużej mocy obliczeniowej i reprezentuje realistyczną górną granicę wydajności takich aplikacji naukowych. Rzeczywista implementacja zależy od kilkunastu parametrów, z których wszystkie mogą mieć znaczący wpływ na wynikową wydajność i dlatego wymagają dopracowania. Poniżej opisujemy parametry HPL, które zostały dostrojone:

1. Rozmiar bloku (NB). Jest określany w odniesieniu do rozmiaru problemu i wydajności podstawowych jąder BLAS. Użyliśmy czterech różnych rozmiarów bloków, a mianowicie 192, 256, 512 i 768.

2. Siatka procesowa ($p \times q$). Jest to liczba wierszy i kolumn procesu w siatce obliczeniowej. Podobnie jak w przypadku większości klastrów, empirycznie zaobserwowaliśmy, że na EC2 lepiej jest używać siatek procesów, gdzie $p \leq q$. Jest to produkt przepływu danych algorytmów używanych w HPL.

3. Algorytm rozgłaszania (BFACT). Zależy to od rozmiaru problemu i wydajności sieci. Testy wykazały, że najlepsze parametry rozgłoszeniowe to 3 i 5. W przypadku dużych maszyn wyposażonych w szybkie węzły w porównaniu z dostępną przepustowością sieci, algorytm 5 jest uważany za najlepszy.

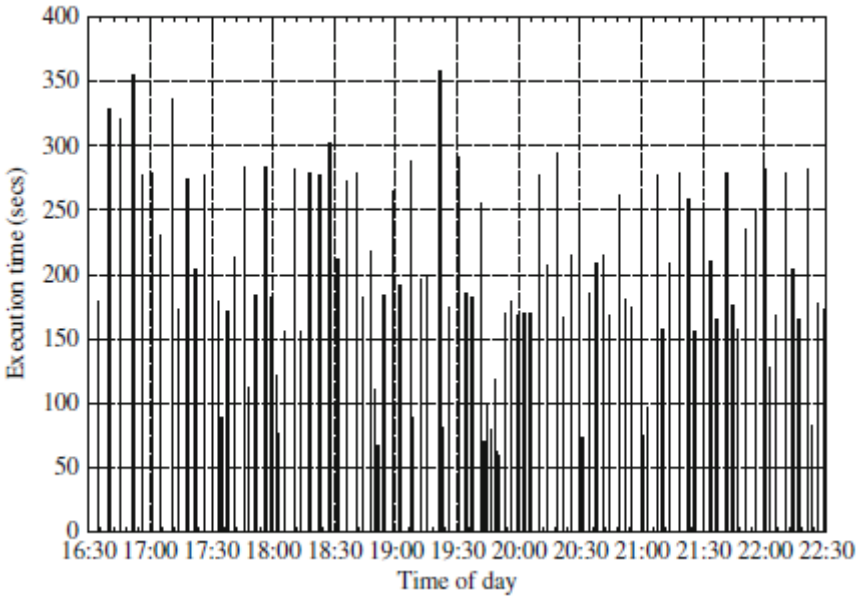
Jeśli chodzi o inne ustawienia HPL, zachowaliśmy je naprawione we wszystkich eksperymentach.

Skalowanie wewnątrzwęzłowe

Naszą empiryczną analizę wydajności EC2 dla algebry liniowej rozpoczynamy od oceny spójności osiągalnej wydajności na pojedynczym węźle. Aby ocenić spójność wydajności dostarczanej przez węzły EC2, wykonaliśmy DGEMM — jądro mnożenia macierzy BLAS — i testy HPL przez 24 godziny, powtarzając eksperyment w różnych dniach. Najpierw skupiamy się na wynikach DGEMM, a następnie omawiamy wyniki z HPL. DGEMM jest rdzeniem biblioteki Basic Linear Algebra Subrouines (BLAS). Implementując podstawową operację mnożenia dwóch macierzy, DGEMM jest budulcem wszystkich innych procedur BLAS poziomu 3 i praktycznie każdego algorytmu algebry liniowej. Jest wysoce zoptymalizowany dla każdej architektury docelowej, a jego wydajność, często w ponad 90% zakresie sprawności, jest zwykle interpretowana jako szczytowa osiągalna wydajność procesora.

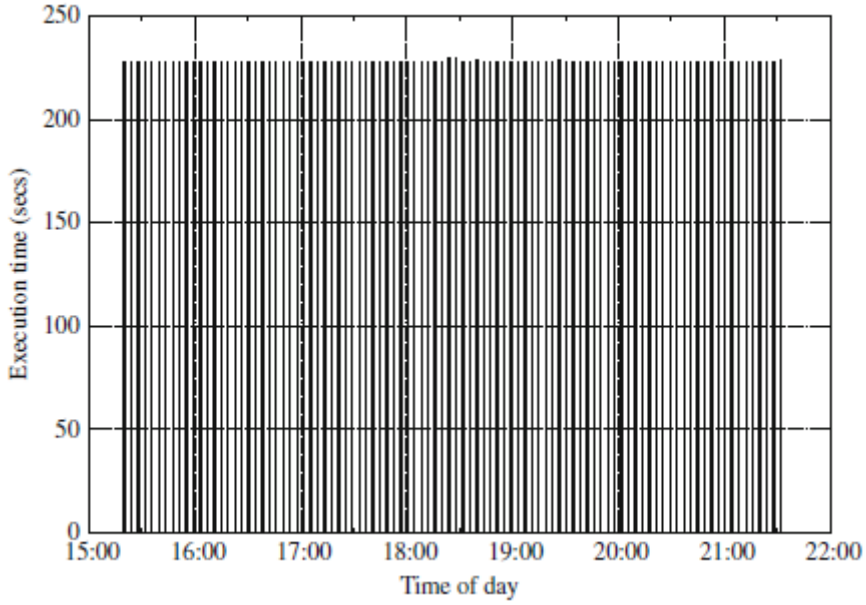
Ocena pojedynczego węzła DGEMM

W naszym eksperymencie DGEMM inicjujemy trzy macierze kwadratowe A, B i C o stałym rozmiarze, a następnie wywołujemy implementację GotoBLAS DGEMM (Goto, 2010). Mierzymy tylko czas wywołania biblioteki BLAS, a nie czas poświęcony na przydzielanie i inicjowanie macierzy. Ze względu na gęsty charakter macierzy biorących udział w eksperymentach – większość wpisów jest niezerowa – oczekujemy niewielkich lub żadnych fluktuacji w czasie wykonania na pojedynczym węźle niezależnie od wielkości problemu i liczby użytych rdzeni. Rysunek przedstawia czas do wykonania tego samego obliczenia DGEMM wielokrotnie w ciągu sześciu godzin na EC2.

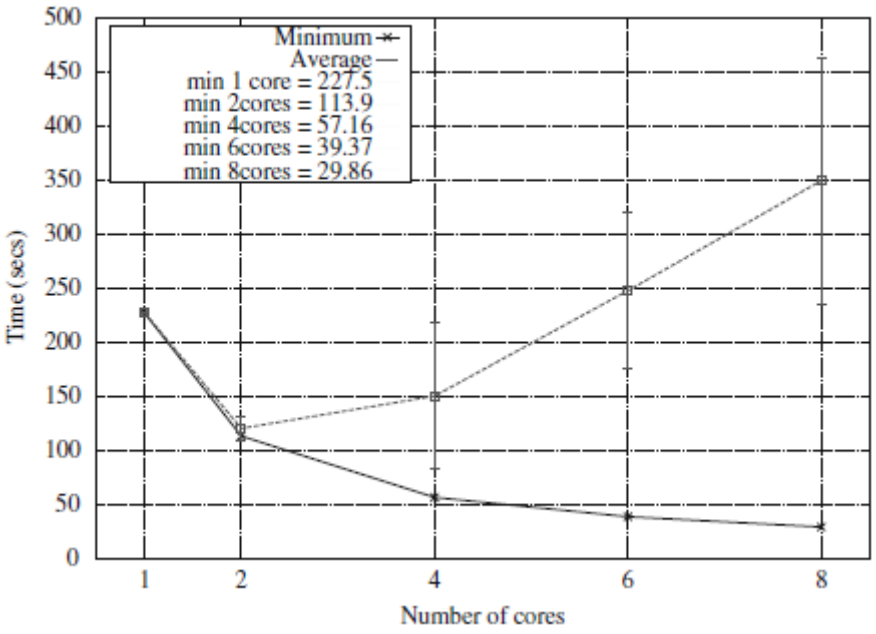


Ze względów kosmicznych skupiamy się na sześciu godzinach; jednak reszta czasu wykazuje podobne zachowanie. W tym eksperymencie wykorzystano tylko cztery z ośmiu rdzeni instancji c1.xlarge. Wyniki pokazują bardzo dużą zmienność czasu wykonania ze średnią 191,8 sekundy i odchyleniem standardowym 68,6 sekundy (36% średniej). Istnieje kilka możliwych źródeł takiej zmienności: (1) proces nie jest uruchamiany przez dłuższe, zmienne okresy czasu, (2) wątki są za każdym razem zaplanowane na różnych rdzeniach (zmniejszając wydajność pamięci podręcznej pierwszego poziomu) oraz (3) pamięć podręczna ostatniego poziomu współdzielona przez wszystkie rdzenie jest mniej dostępna dla każdego wątku (ponieważ jest używana przez inny wątek na innym rdzeniu). Rysunek

pokazuje eksperyment podobny do rysunku wcześniejszego, ale z użyciem tylko jednego z ośmiu rdzeni.



Eksperyment nie wykazuje żadnej zmienności przy użyciu tylko jednego rdzenia, który jest obecny przy większej liczbie rdzeni. Przy średnim czasie wykonania 227,9 s odchylenie standardowe wynosi tylko 0,23 s. Wyniki dla wszystkich ośmiu rdzeni wykazują jeszcze wyższy poziom zmienności niż pierwszy rysunek. Zmniejszona zmienność pojedynczego rdzenia pokazuje, że przyczyna (1) powyżej jest mało prawdopodobna. Proces jest zaplanowany podobnie, ale wydajność jest znacznie bardziej przewidywalna. Nie można jednak wykluczyć przyczyny (2), ponieważ Amazon EC2 nie zapewnia mechanizmu przypinania wątków do poszczególnych rdzeni, dzięki czemu wątek jest zawsze wykonywany na tym samym fizycznym rdzeniu procesora. Wreszcie, przypuszczamy, że przyczyna (3) odgrywa co najmniej tak samo istotną rolę jak (2) i przyglądamy się różnym eksperymentom, aby zbadać te efekty. Łatwo zaobserwować efekty zastosowania różnej liczby rdzeni. Średni i minimalny czas wykonania w stosunku do liczby rdzeni przedstawiono tu.



Na tym wykresie DGEMM był wykonywany przez kilka godzin na macierzy o rozmiarze n = 10 k, więc macierze (każda po 762 MB) nie mogą zmieścić się w 8 MB pamięci podręcznej ostatniego poziomu procesora Intel XEON z instancją c1.xlarge. Przedstawione wyniki to średnie i minimalne czasy wykonania dla DGEMM. Ponownie alokacja i inicjalizacja pamięci nie są uwzględniane w taktowaniu. Słupki błędów pokazują odchylenie standardowe średniej. Jest kilka godnych uwagi cech wykresu:

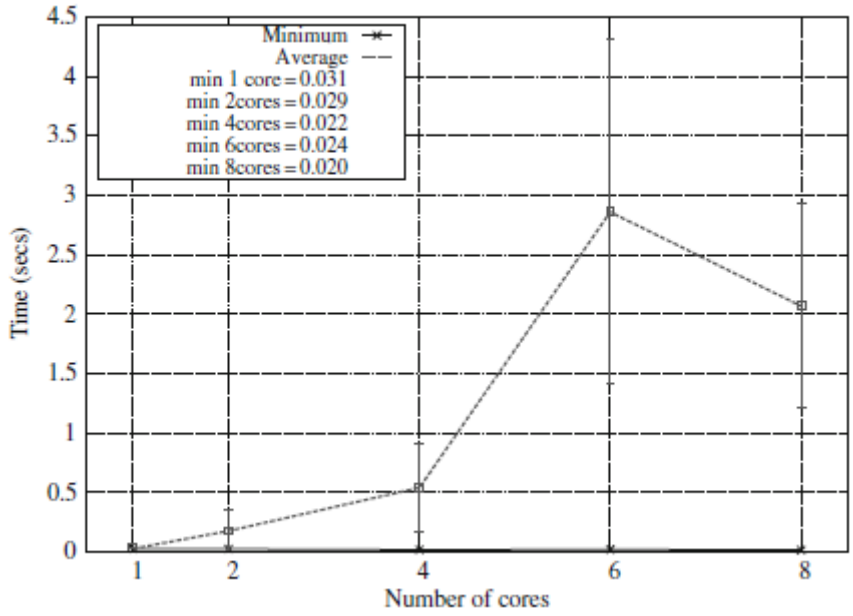
1. Najlepsza wydajność na wykresach (linia minimum) pokazuje, że możemy osiągnąć około 90% wydajności. Taka wydajność jest zbliżona do optymalnej osiągalnej. Na przykład taka wydajność byłaby oczekiwana w samodzielnym węźle. Zauważamy, że sama wirtualizacja wyraźnie nie ma znaczącego wpływu na szczytową wydajność.

2. Średnia wydajność pokazuje, że podobne obliczenia prawdopodobnie nigdy nie osiągną optymalnej wartości. Średnia wydajność - wynikająca z odwrotności rozpiętości na wykresie między min i średnią - dramatycznie spada. Najlepsza średnia wydajność na dwóch rdzeniach jest wciąż kilkakrotnie gorsza niż minimalny czas wykonania na ośmiu rdzeniach.

3. Wraz ze wzrostem liczby rdzeni, średnia znacznie wzrasta wraz z odchyleniem standardowym. Odchylenie standardowe wzrasta o cztery rzędy wielkości. Oczekiwana wydajność tak znacznie odbiega od najlepszej wydajności, że przy użyciu ośmiu rdzeni rzadko można oczekiwać osiągnięcia najlepszej wydajności.

4. Niedostateczne wykorzystanie to dobra polityka wobec EC2. Użycie dwóch rdzeni wydaje się być najlepszym punktem w tym eksperymencie. Wydajność jest dobra, a średnia i oczekiwana wydajność najlepsza i zgodna z optymalną. Używanie czterech rdzeni już trwa średnio dłużej niż użycie dwóch, chociaż użycie czterech jest nadal nieco szybsze niż pojedynczego rdzenia, a trend pogarsza się tylko wraz ze wzrostem liczby rdzeni. Pamiętaj, że najszybszy oczekiwany czas jest uzyskiwany przy użyciu tylko jednej czwartej maszyny!

Chociaż możemy przypisać pogorszenie wydajności przy użyciu więcej niż dwóch rdzeni na węzeł na coraz gorszemu zachowaniu pamięci podręcznej, nie możemy łatwo odróżnić, które pamięci podręczne są pomijane. W procesorach wielordzeniowych w tych przypadkach rdzenie mają indywidualne pamięci podręczne i dużą współużytkowaną pamięć podręczną ostatniego poziomu (LLC). Przypięcie wątku do konkretnego rdzenia pomogłoby wątkowi utrzymać spójność pamięci podręcznej dla poszczególnych pamięci podręcznych, podczas gdy zmniejszenie zapotrzebowania wątków na pamięć zmniejszyłoby rywalizację o współdzieloną LLC. Jądra dostarczane przez Amazon EC2 nie zapewniają jednak obsługi wątków. Aby spróbować odróżnić te efekty bez możliwości naprawiania wątków na poszczególnych rdzeniach, przeprowadziliśmy podobne eksperymenty DGEMM z mniejszymi macierzami, które pasują do LLC procesora. Rysunek pokazuje średni i minimalny czas wykonania przy użyciu różnej liczby rdzeni dla DGEMM na małych macierzach n = 500, co daje macierze 1,9 MB, które można łatwo zmieścić w 8 MB LLC.



Średnie słupki błędów czasu wykonania dają odchylenie standardowe od średniej. Ponownie alokacja i inicjalizacja pamięci nie są uwzględniane w taktowaniu. Zwracamy uwagę na kilka cech wykresu:

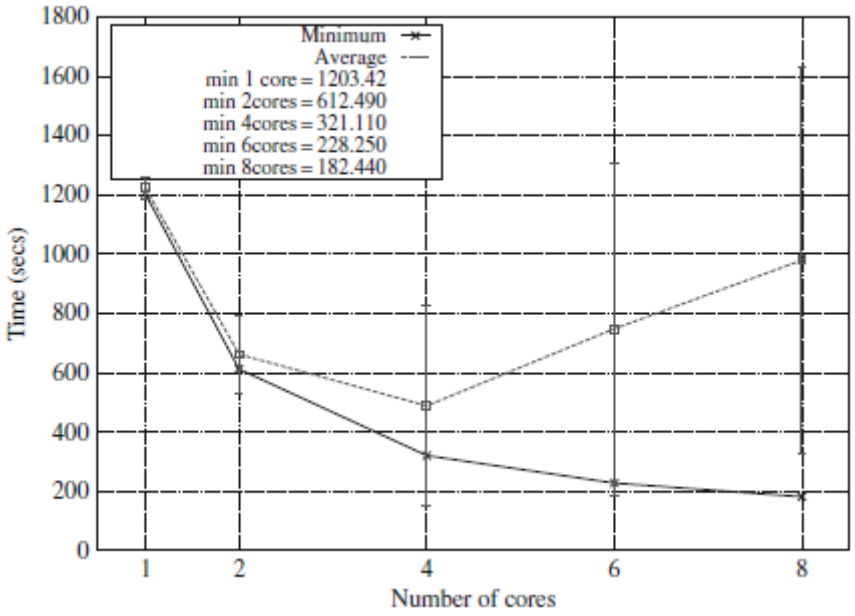
1. Podobnie jak w przypadku modułu DGEMM działającego poza pamięcią podręczną, minimalne czasy wykonania są zgodne z oczekiwaną wydajnością węzła autonomicznego.

2. Rywalizacja prawdopodobnie dotyczy pamięci podręcznej ostatniego poziomu. Dwa rzędy wielkości spadku wydajności między pojedynczym a ośmioma rdzeniami to także różnica w czasie między dostępem do pamięci podręcznej ostatniego poziomu a pamięcią główną.

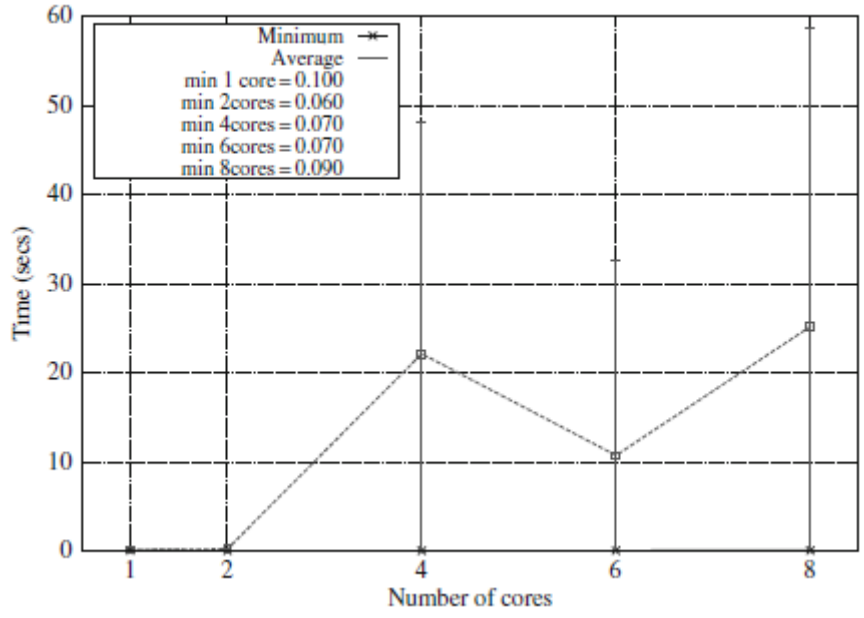
3. Możliwość przypinania gwintów do rdzeni prawdopodobnie by znacząco pomogła. Korzystanie z wielu rdzeni zawsze działa gorzej niż z pojedynczym rdzeniem, co oznacza znaczne obciążenie koordynacji. Ponieważ minimum dobrze się skaluje, przypuszczamy, że średnia cierpi z powodu złego zachowania pamięci podręcznej. Jednak średni czas z wieloma rdzeniami nie staje się o rząd wielkości gorszy – co mogłoby być spowodowane przejściem do pamięci głównej – dopóki wszystkie rdzenie nie zostaną wykorzystane. Błędy w pamięci podręcznej niższego poziomu są zatem bardziej prawdopodobne, gdy instancja z mniej niż ośmioma rdzeniami nie zostanie w pełni wykorzystana.

Ocena pojedynczego węzła HPL

Aby określić, czy efekty, które zaobserwowaliśmy przy użyciu skali DGEMM do wielu węzłów, stosujemy benchmark HPL, który rozwiązuje układ równań liniowych za pomocą faktoryzacji LU. HPL można skalować do dużych siatek obliczeniowych. Jednak najpierw rozważamy HPL, mając podobne macierze do poprzednich eksperymentów DGEMM, a mianowicie rozwiązywanie układu równań liniowych na pojedynczym węźle, w którym dane pasują i nie pasują do LLC procesora. W następnej sekcji rozszerzymy HPL, aby zbadać skalowanie międzywęzłowe za pomocą algorytmów równoległych. Na rysunku wielokrotnie wykonujemy HPL z n = 25 k na instancji Amazon EC2 c1.xlarge.



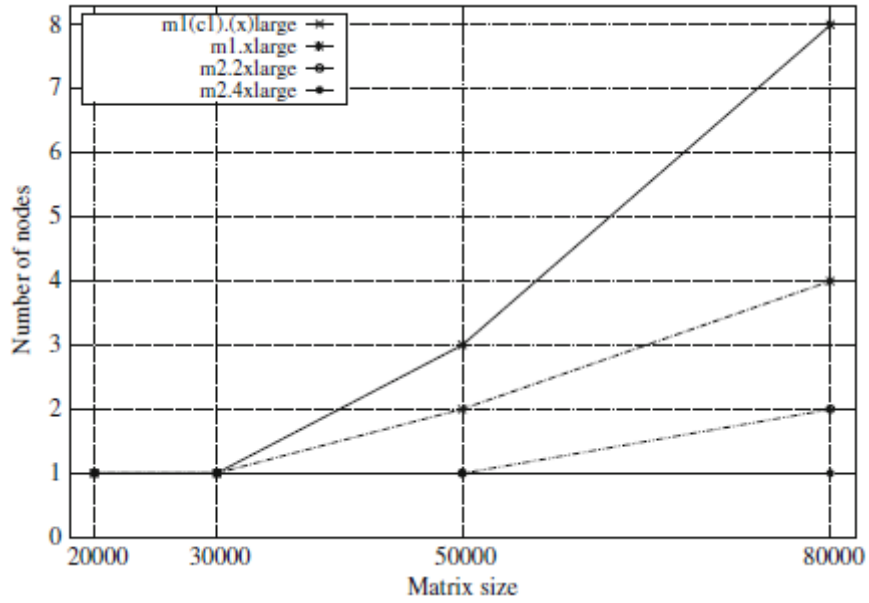
Wykreślamy średnie i minimalne czasy wykonania w stosunku do liczby rdzeni. Podobnie jak w przypadku eksperymentów DGEMM, słupki błędów pokazują jedno odchylenie standardowe. Podobnie jak w pierwszym eksperymencie DGEMM, te macierze nie mieszczą się w LLC procesora, ale mieszczą się w głównej pamięci pojedynczej instancji. Wyniki są dość podobne do DGEMM, ale pokazują nieco inne trendy. Nie porównujemy bezpośrednio eksperymentów DGEMM i HPL ze względu na użycie bardzo różnych algorytmów. Zwracamy tylko uwagę, że trendy wykazują podobne zachowanie na jednym węźle. Na ostatnim rysunku pojedynczego węzła pokazujemy HPL z macierzą o rozmiarze n = 1 k, tak aby wszystkie dane do obliczeń mieściły się w LLC. Rysunek przedstawia średni (i jedno odchylenie standardowe) i minimalne czasy wykonania powtarzanych przebiegów testu porównawczego HPL.



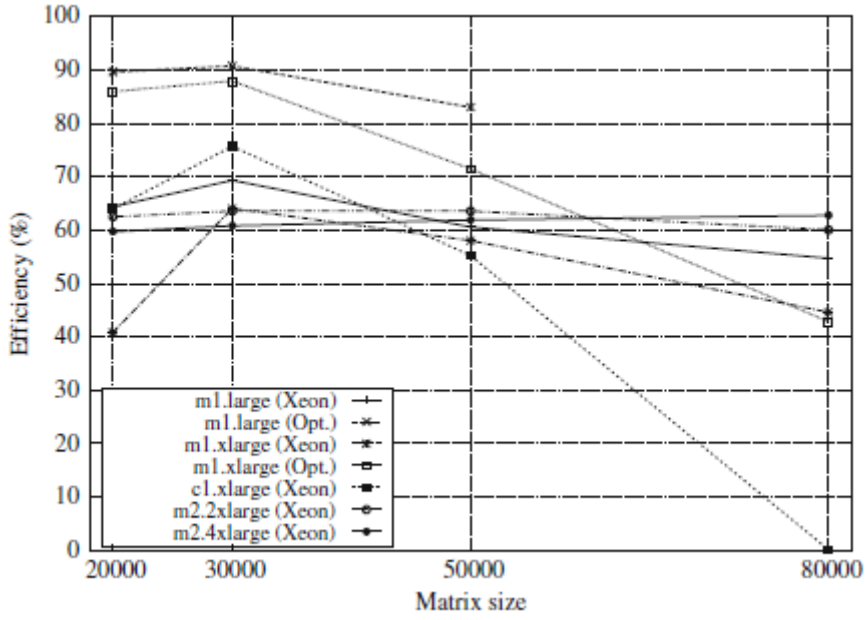
Wyniki pokazują podobne zachowanie do odpowiedniego eksperymentu DGEMM. Nie mogliśmy znaleźć źródła anomalnej poprawy wydajności przy sześciu rdzeniach w porównaniu z czterema lub ośmioma. W następnej sekcji rozszerzymy HPL na wiele węzłów, aby przyjrzeć się wpływowi rywalizacji na równoległe obliczenia w komercyjnym środowisku chmury. Z eksperymentów na jednym węźle wnioskujemy, że bardzo wysoka wariancja wydajności oznacza, że najlepsza osiągnięta wydajność nie jest dobrą miarą oczekiwanej wydajności. Średnia oczekiwana wydajność może być o kilka rzędów wielkości gorsza w zależności od zachowania pamięci podręcznej algorytmu zarówno pod względem wydajności, jak i czasu do rozwiązania. W naszych eksperymentach najlepszą średnią oczekiwaną wydajność uzyskuje się na Amazon przy użyciu znacznie mniejszej ilości maszyny niż jest przydzielone: W eksperymencie z dostępem do dużej ilości pamięci głównej, użycie tylko jednej czwartej maszyny zapewniło najlepszą oczekiwaną wydajność!

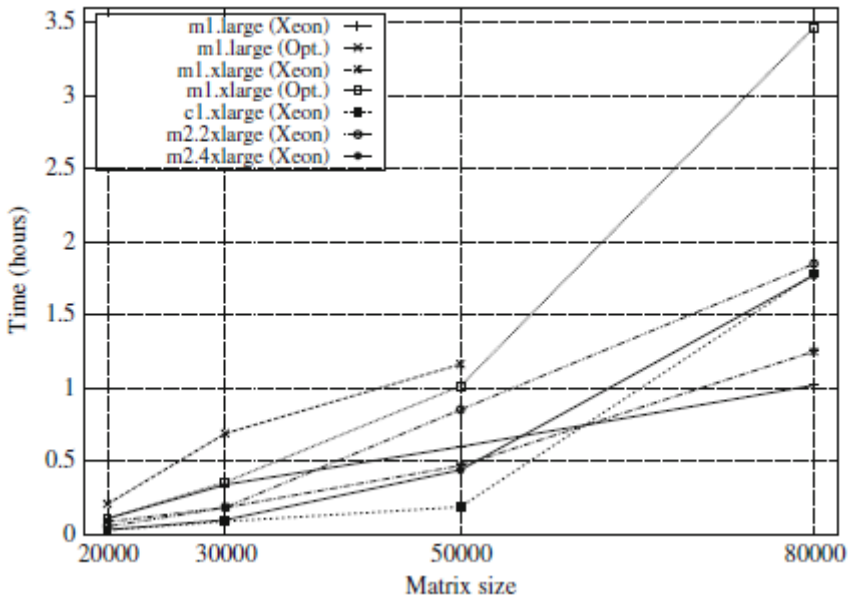
Skalowanie międzywęzłowe

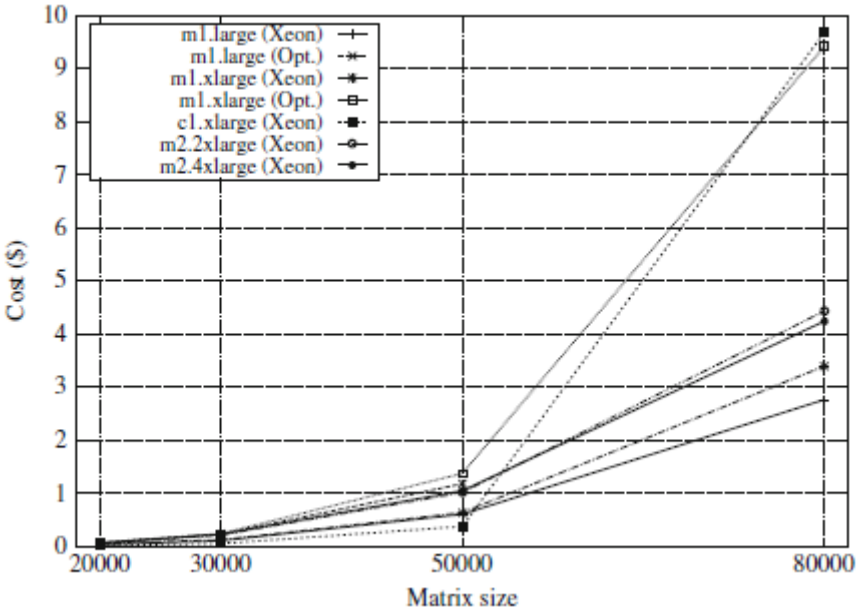
W poprzedniej sekcji przedstawiono znaczący wpływ rywalizacji na wydajność pojedynczego węzła w komercyjnym środowisku chmury. W tej sekcji rozszerzamy naszą analizę empiryczną o równoległe algorytmy wielowęzłowe przy użyciu testu porównawczego HPL. Test porównawczy HPL reprezentuje ściśle powiązane, wysoce równoległe algorytmy często używane w zastosowaniach naukowych. Korzystanie z różnych typów instancji pozwoliło nam zobaczyć, jaki wpływ na wydajność, wydajność i koszt ma zróżnicowana ilość dostępnej pamięci RAM i wielkość problemu. Aby obliczyć całkowitą liczbę rdzeni użytych do konkretnego wykonania, użyj wyniku p×q× Threads. Ze względu na liczne parametry HPL konieczne było wypróbowanie różnych konfiguracji wejściowych, aby zmaksymalizować wydajność. Aby zmaksymalizować wydajność, zminimalizowaliśmy zależność od połączenia, maksymalizując przydział pamięci na węzeł używany do rozwiązania problemu o określonym rozmiarze. Rysunek pokazuje liczbę węzłów używanych dla różnych typów instancji w celu rozwiązania każdego rozmiaru problemu.

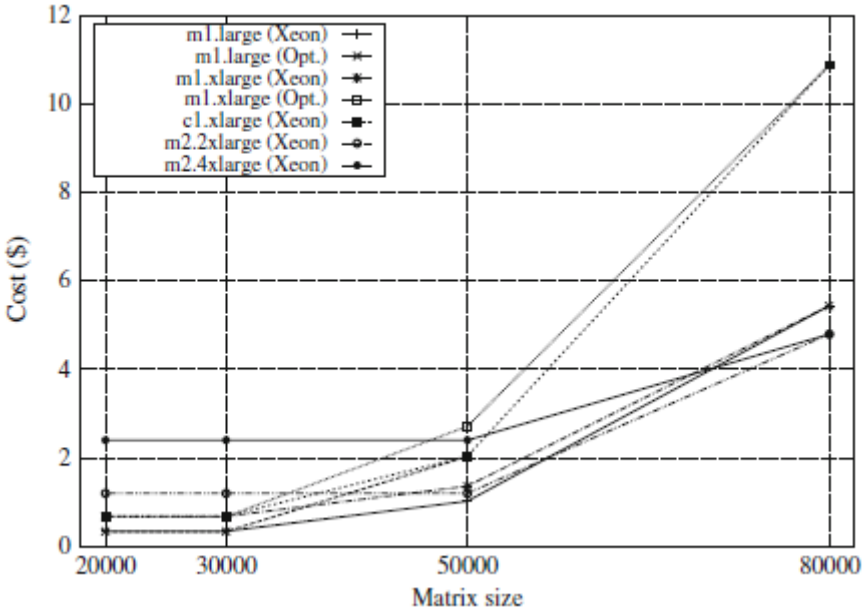


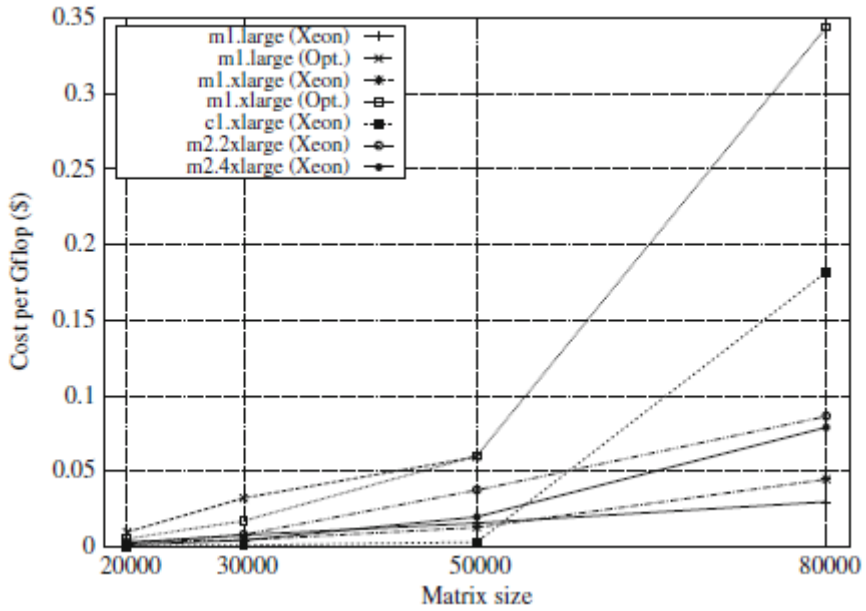
Tylko instancja m2.4xlarge z 68,4 GB pamięci RAM jest wystarczająco duża, aby rozwiązać wszystkie rozmiary problemów na jednym węźle. Wykorzystaliśmy to jako punkt odniesienia na ryc. 21.8, aby ograniczyć wpływ wykorzystania sieci na wydajność. Na poniższych rysunkach ryc. 21.8, 21.9, 21.10, 21.11, 21.12 i 21.13 przedstawiamy różne aspekty wyników naszych eksperymentów porównawczych HPL.

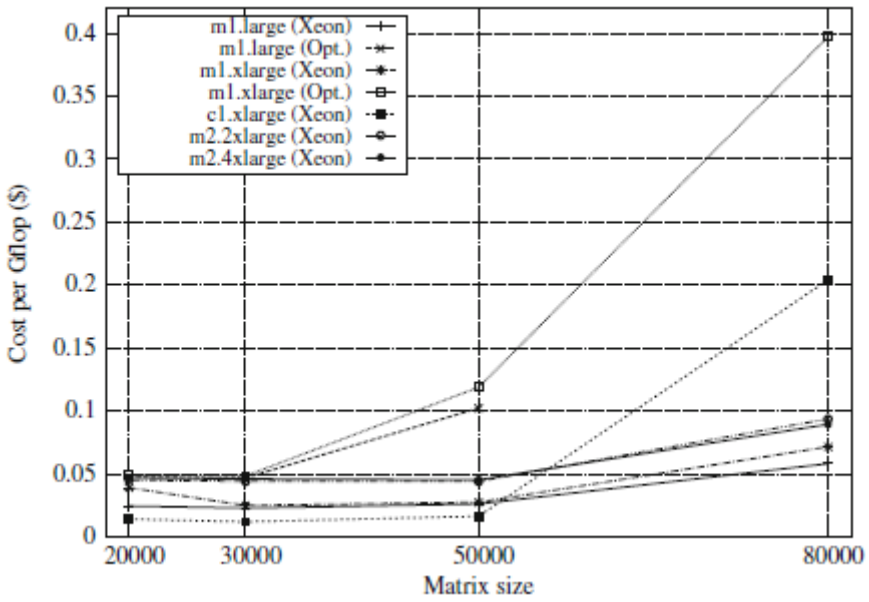

Efficiency (%)
Matrix size
m1.large (Xeon)
m1.large (Opt.)
m1.xlarge (Xeon)
m1.xlarge (Opt.)
c1.xlarge (Xeon)
m2.2xlarge (Xeon)
m2.4xlarge (Xeon)
0
10
20
30
40
50
60
70
80
90
100
20000
30000
50000
80000


Time (hours)
Matrix size
m1.large (Xeon)
m1.large (Opt.)
m1.xlarge (Xeon)
m1.xlarge (Opt.)
c1.xlarge (Xeon)
m2.2xlarge (Xeon)
m2.4xlarge (Xeon)
0
0.5
1
1.5
2
2.5
3
3.5
20000
30000
50000
80000


m1.large (Xeon)
m1.large (Opt.)
m1.xlarge (Xeon)
m1.xlarge (Opt.)
c1.xlarge (Xeon)
m2.2xlarge (Xeon)
m2.4xlarge (Xeon)
Cost ($)
Matrix size
0
1
2
3
4
5
6
7
8
9
10
20000
30000
50000
80000


m1.large (Xeon)
m1.large (Opt.)
m1.xlarge (Xeon)
m1.xlarge (Opt.)
c1.xlarge (Xeon)
m2.2xlarge (Xeon)
m2.4xlarge (Xeon)
Cost ($)
Matrix size
0
2
4
6
8
10
12
20000
30000
50000
80000





Najpierw omawiamy najlepsze wyniki uzyskane w celu uzyskania rozsądnej dolnej granicy wydajności w kontrowersyjnym komercyjnym środowisku chmury. W tych równoległych eksperymentach czasy minimalne nie różnią się tak bardzo od średniej, jak w przypadku eksperymentów z jednym węzłem, z kilku powodów: (1) wymagany czas (ponad godzina w przypadku dużych problemów) implikuje efekt uśredniania i (2) obciążenie dla zadań równoległych jest wyższe niż w przypadku eksperymentów z jednym węzłem ze względu na wykorzystanie sieci. Na koniec zauważamy, że na tych rysunkach linia dla instancji m1.large korzystających z procesora Opteron nie rozciąga się do n = 80 k, ponieważ nie mogliśmy przydzielić wystarczającej liczby instancji m1.large z procesorami Opteron w jednej strefie dostępności, aby rozwiązać problem rozmiar n = 80 tys.

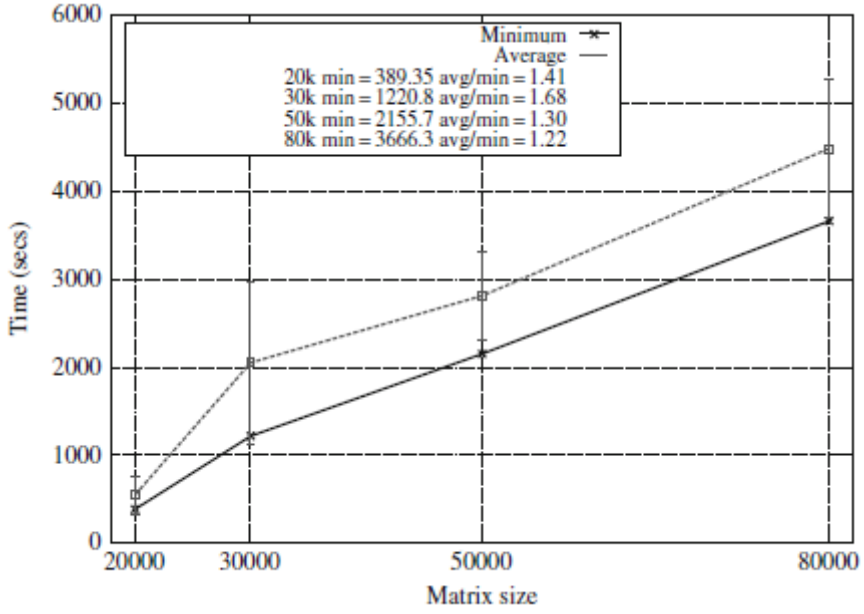
Minimalna ocena HPL

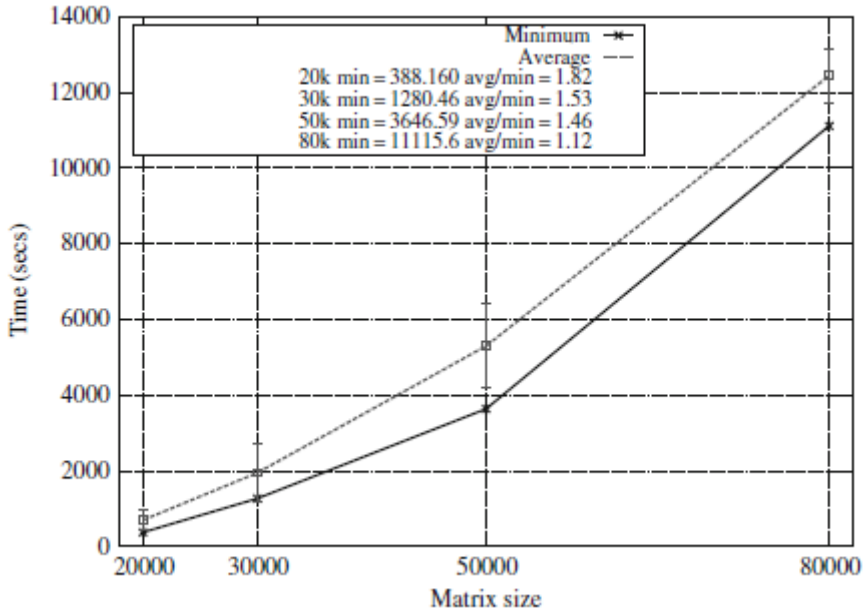
Pierwszy z nich rysunek pokazuje, że wydajność jest ogólnie lepsza niż 60% dla problemów o rozmiarach poniżej 80 tys. Tutaj wydajność jest uważana za procent szczytowej wartości teoretycznej pomnożony przez liczbę wykorzystywanych węzłów. Uważamy, że 60% wydajności jest rozsądną

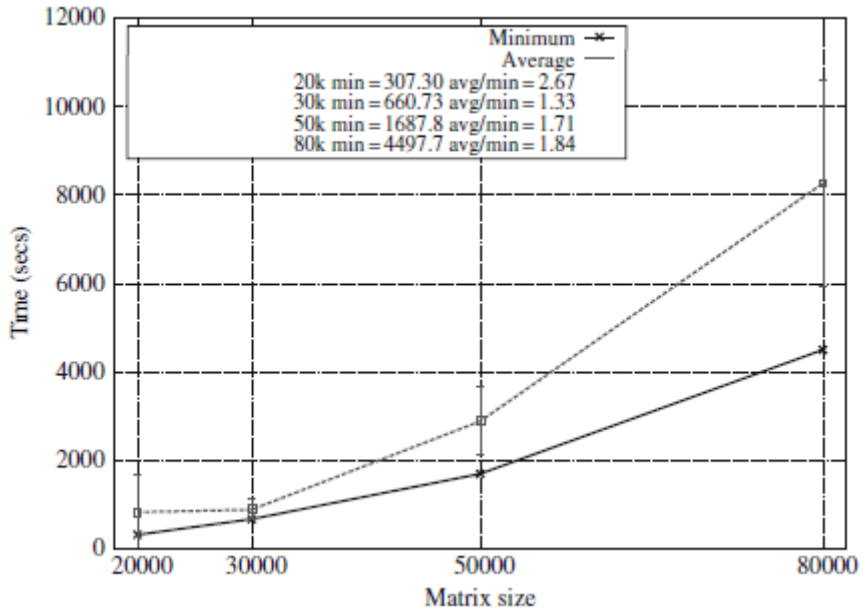
wydajnością, biorąc pod uwagę stosunkowo powolne połączenie zapewniane przez Amazon EC2 w porównaniu do specjalnie zbudowanych systemów HPC. Bilans zasobów dostępnych do obliczeń jest wyraźnie ważny. Na przykład wystąpienia m2.2xlarge i m2.4xlarge mają mniej więcej taką samą wydajność przy 80 tys., chociaż HPL potrzebuje dwóch wystąpień m2.2xlarge i tylko jednego m2.4xlarge. W tym przypadku połączenie nie ma znaczącego wpływu, ponieważ węzły są wyposażone w wystarczającą ilość pamięci RAM, aby utrzymać zajętość procesora. Jednak instancje c1.xlarge cierpią na poważne pogorszenie wydajności. Przy n = 80 k instancje c1.xlarge wykonują dwa rzędy wielkości gorzej niż pojedynczy węzeł. Przypuszczamy, że 7 GB pamięci RAM węzłów c1.xlarge jest niewystarczające, aby utrzymać obciążenie procesora, biorąc pod uwagę to samo połączenie między instancjami. Ogólnie rzecz biorąc, węzły z większą ilością pamięci RAM znacznie lepiej skalują rozmiar problemu, niż oczekiwano, ze względu na wysoki stosunek wymaganej liczby operacji do rozmiaru danych do rozwiązania gęstego systemu liniowego. Chociaż wydajność jest ważna dla oceny skalowalności implementacji, w przypadku każdego konkretnego eksperymentu najważniejszym problemem jest zazwyczaj czas na rozwiązanie. Drugi rysunek przedstawia łączny czas rozwiązania dla różnych typów instancji według wielkości problemu. Ten wykres pokazuje, że chociaż wydajność większych węzłów jest nieco lepsza niż mniejszych, w większości przypadków mniejsze węzły szybciej osiągają rozwiązanie. Wystąpienia c1.xlarge, zalecane dla aplikacji o dużej mocy obliczeniowej, oraz wystąpienia m2.2xlarge i m2.4xlarge, zalecane dla aplikacji o dużej ilości pamięci, są na ogół najwolniejsze. Jednym z ważnych problemów podczas korzystania z komercyjnych środowisk chmurowych są różne koszty różnych instancji. Trzeci rysunek przedstawia koszt rozwiązania dla różnych typów instancji według rozmiaru problemu. Na tym wykresie koszt jest proporcjonalny do drugiego, aby zilustrować koszty krańcowe poniesione podczas przydzielania klastra do rozwiązania kilku problemów. Czwarty rysunek przedstawia porównywalne rzeczywiste koszty korzystania z różnych typów instancji w celu rozwiązania pojedynczego wykonania każdego rozmiaru problemu; czyli ten wykres zawiera koszty pozostałej godziny po zakończeniu realizacji. Istotną różnicą między figurami jest stosunek największych węzłów do mniejszych węzłów. Stosując koszt bezwzględny, największe węzły są najtańsze w przypadku dużych problemów, ale przy zastosowaniu kosztów proporcjonalnych są droższe. Ponieważ trendy proporcjonalne dostarczają więcej informacji dla różnych rozmiarów problemów i dla wielu zadań, wnioskujemy, że mniejsze wystąpienia m1.large i m1.xlarge są najbardziej opłacalne w przypadku zadań równoległych. Oprócz kosztu do rozwiązania, bierzemy również pod uwagę bardziej ogólną miarę kosztu, $/GFLOP, obliczoną na podstawie stosunku całkowitej wartości GFLOPS zwróconej przez test porównawczy HPL do (proporcjonalnego) kosztu określonego obliczenia. Miara ta umożliwia przybliżoną konwersję oczekiwanych kosztów dla innych rozmiarów problemów, w tym innych zastosowań naukowych charakteryzujących się podobnymi potrzebami obliczeniowymi. Ostatnie dwa rysunki pokazują wyniki testów porównawczych HPL, podając koszt na GFLOP dla różnych instancji według rozmiaru problemu. Koszt na miarę GFLOP powiększa różnice między typami instancji, ale oczywiście najlepszy typ instancji– m1.large – to nadal najlepszy stosunek ceny do wydajności. Instancja m1.large jest najszybsza według drugiego rysunku i najlepsza pod względem stosunku ceny do wydajności, co prowadzi nas do rekomendacji najmniejszej (64-bitowej) instancji dla większości zadań obliczeń równoległej algebry liniowej w środowisku chmury Amazon EC2 według do empirycznej górnej granicy wydajności. Instancje o dużej mocy obliczeniowej i dużej ilości pamięci nie są warte dodatkowych kosztów w naszych eksperymentach. Biorąc pod uwagę dużą zmienność wydajności pojedynczych węzłów, w poniższej sekcji zbadamy oczekiwaną wydajność z instancji zamiast górnej granicy, aby określić, czy nasze zalecenie dotyczy również oczekiwanej średniej wydajności.

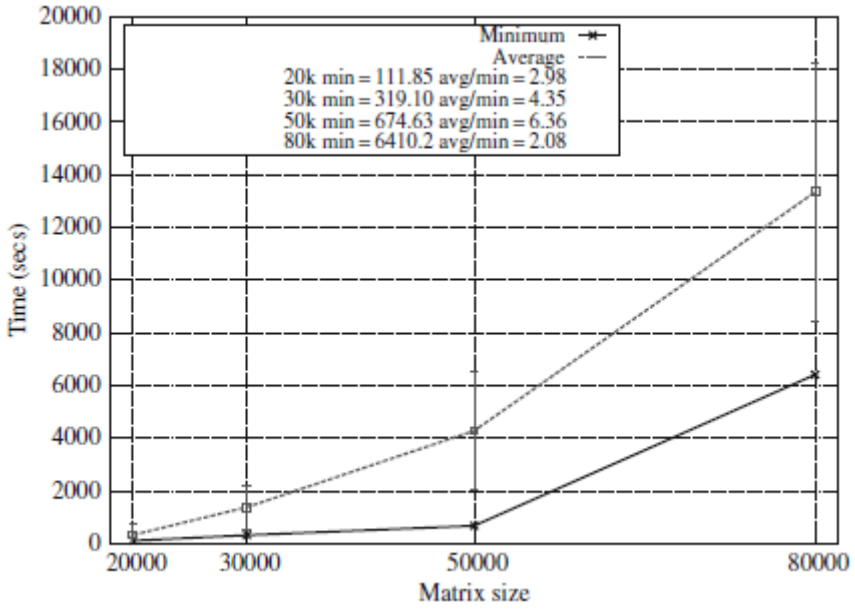
Średnia ocena HPL

Minimalne czasy wykonania z poprzedniej sekcji zapewniają zgrubną górną granicę wydajności. W tej sekcji zbadamy średnie czasy wykonania dla HPL, aby zapewnić lepsze oszacowanie oczekiwanej wydajności aplikacji z gęstą algebrą liniową. Cztery kolejne rysunki przedstawiają minimalny i średni (oraz odchylenie standardowe) czasy wykonania dla różnych instancji według rozmiaru problemu.









Nie pokazujemy instancji m2.2xlarge lub m2.4xlarge z powodu niewystarczających danych. Te duże instancje są również dość drogie do przydzielenia. Podobnie jak w przypadku eksperymentów na jednym węźle, Rysunki te pokazują, że oczekiwana wydajność jest gorsza od najlepszej wydajności, ale ze znacznie mniejszym marginesem. Rzeczywiście, dla instancji m1.large na pierwszym rysunku oczekiwana wydajność przy n = 80 k jest tylko o 22% gorsza niż najlepszy czas wykonania. Ogólnie rzecz biorąc, średnia dla eksperymentów HPL jest od 30% do 2 razy gorsza niż minimalne czasy wykonania. Jak wspomnieliśmy na początku tej sekcji, uważamy, że mniejsze różnice między oczekiwanym czasem wykonania a najlepszymi czasami wynikają z uśrednionego wpływu długości eksperymentów i większego narzutu testu porównawczego HPL z ruchu sieciowego (w porównaniu do eksperymentów DGEMM). Wystąpienie m1.large - najmniejsze, które testowaliśmy - pozostaje najlepszą instancją w EC2 dla ściśle powiązanych, intensywnych obliczeń i równoległych zadań. Chociaż średni oczekiwany czas jest dłuższy od minimum o 20% dla największego rozmiaru problemu, oczekiwany czas jest nadal szybszy niż następny najszybszy czas minimalny (m1.xlarge Xeon). Biorąc pod uwagę, że m1.large również kosztuje o połowę mniej niż m1.xlarge, najmniejsza instancja jest wyraźnym zwycięzcą w

przypadku mocno sprzężonej, gęstej algebry liniowej, którą oszacowaliśmy, oraz wysokich kosztów krańcowych dla obliczeń o dużej mocy obliczeniowej lub węzły pamięci nie są opłacalne.

Wnioski

Wykazaliśmy empirycznie wydajność obliczeniową wysokowydajnych aplikacji numerycznych w komercyjnym środowisku chmury, gdy zasoby są współdzielone w warunkach dużej rywalizacji. Poprzez studium przypadku z wykorzystaniem testu porównawczego Linpack pokazujemy, że wykorzystanie pamięci podręcznej staje się wysoce nieprzewidywalne i podobnie wpływa na czas obliczeń. W przypadku niektórych problemów nie tylko bardziej wydajne jest niepełne wykorzystanie zasobów, ale rozwiązanie można znaleźć wcześniej w czasie rzeczywistym (czas na ścianach). Pokazujemy również, że najmniejsza, najtańsza (64-bitowa) instancja na Amazon. Komercyjne środowisko chmurowe EC2 jest nie tylko najszybsze, ale także najlepsze pod względem stosunku ceny do wydajności. Przedstawiliśmy średnią oczekiwaną wydajność i czas wykonania, oczekiwany koszt do ukończenia oraz miary wariancji - tradycyjnie ignorowane w kontekście obliczeń o wysokiej wydajności - w celu określenia wydajności i wydajności komercyjnego środowiska chmury Amazon EC2. W warunkach wszechobecnej rywalizacji o zasoby oczekiwana wydajność w środowisku chmury różni się o rząd wielkości od najlepszej osiąganej wydajności. Dochodzimy do wniosku, że istnieje znaczna przestrzeń do poprawy w zapewnianiu przewidywalnej wydajności w takich środowiskach.