Prace Instytutu Mechaniki Górotworu PAN
Tom 17, nr 1-2, czerwiec 2015, s. 41-48

Zastosowanie systemów rozproszonych w wizualizacjach 3D

Magdalena Bundyra

Instytut Mechaniki Górotworu PAN; ul. Reymonta 27, 30-059 Kraków
AGH Akademia Górniczo-Hutnicza, Al. Adama Mickiewicza 30, 30-059 Kraków

Jerzy Wiciak

AGH Akademia Górniczo-Hutnicza, Al. Adama Mickiewicza 30, 30-059 Kraków

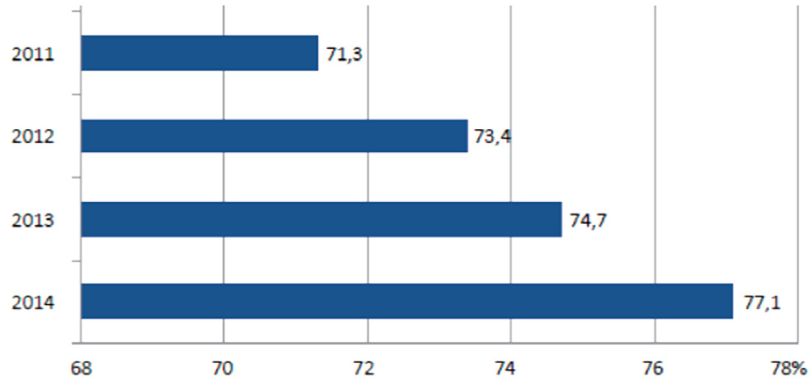
Streszczenie

W artykule zaprezentowana została metodyka projektowania oraz implementacji rozproszonego systemu obliczeniowego na przykładzie obliczeń animacji 3D w programie Blender. Zaproponowany system w założeniach ma być uniwersalny, skalowalny oraz niewymagający konfiguracji przez użytkownika. Jego działanie ma opierać się na nieodpłatnym wykorzystaniu udostępnionych zasobów obliczeniowych. System zaprojektowano w oparciu wykorzystanie darmowych narzędzi opartych na licencji GNU GPL oraz darmowego systemu operacyjnego Linux. Zaproponowane w artykule rozwiązanie stanowi alternatywę dla komercyjnych, zamkniętych rozwiązań.

Słowa kluczowe: farmy obliczeniowe, obliczenia rozproszone, rendering, animacja

1. Wstęp

Według danych Głównego Urzędu Statystycznego z 2014 r. ponad 77% gospodarstw domowych w Polsce posiada przynajmniej jeden komputer (Rys. 1). Mimo rosnącej mocy obliczeniowej komputerów, najczęściej wykorzystywane są one do przeglądania stron internetowych, sprawdzania poczty elektronicznej oraz wykonywania prostych manipulacji na plikach [1].



Rys. 1. Dane Głównego Urzędu Statystycznego dot. ilości komputerów w gospodarstwach domowych

W drugiej połowie lat 90. XX wieku wzrosło zapotrzebowanie na coraz to bardziej skomplikowane obliczenia. Niewystarczające moce obliczeniowe ówczesnych komputerów okazały się impulsem do poszukiwania innych rozwiązań. Aby sprostać rosnącym wymaganiom rozpoczęto przeprowadzanie obliczeń

w trybie rozproszonym. Oznaczało to współpracę wielu maszyn jednocześnie pracujących nad jednym problemem obliczeniowym, w rezultacie skracając czas obliczeń lub poprawiając ich jakość w przypadku obliczeń iteracyjnych.

Na początku XXI wieku procesory jednordzeniowe zostały stopniowo zastępowane procesorami dwurdzeniowymi. Wiązano bardzo duże nadzieje ze zwiększaniem ilości rdzeni. Obowiązujące wtedy prawo Moore'a określało prawidłowość projektowania i wytwarzania procesorów: co 12-18 miesięcy podwaja się liczba tranzystorów w procesorach (prawo to później stosowano w odniesieniu do ilości rdzeni procesora, taktowania zegara oraz pamięci RAM). Głównym problemem okazała się nieliniowa charakterystyka wydzielanego ciepła oraz jego odprowadzenie z układu [2].

Obliczenia prowadzone w trybie rozproszonym polegają na współdzieleniu zasobów obliczeniowych z uwzględnieniem rozproszenia geograficznego. Aby obliczenia w trybie rozproszonym mogły być przeprowadzone, konieczne jest, aby użytkownicy udostępnili moc obliczeniową swoich komputerów (czasami z korzyścią finansową), połączenie komputerów w sieci lokalnej lub globalnej oraz zainstalowanie odpowiedniego oprogramowania. Następnie może rozpocząć się praca w systemie rozproszonym, przy czym jeden z komputerów musi pracować w trybie *master*, a pozostałe w trybie *slave*.

Obliczenia w trybie rozproszonym mają wiele zalet w porównaniu do obliczeń wykonywanych na jednym komputerze, nawet wieloprocesorowym. Można do nich zaliczyć przyśpieszenie obliczeń, zwiększenie niezawodności działania, zwiększenie stopnia wykorzystania mocy obliczeniowej oraz uzyskanie rozwiązania, które może być zbyt duże dla pojedynczej maszyny. Dodatkowo, obliczenia rozproszone niejednokrotnie są jedynym rozwiązaniem na uzyskanie rozwiązania numerycznego w akceptowalnym czasie krótszym niż horyzont symulacji. Są one powszechnie wykorzystywane przez naukowców różnych dziedzin (np. modelowanie klimatu, modelowanie przepływów turbulentnych, obliczenia w astronomii – projekt SETI).

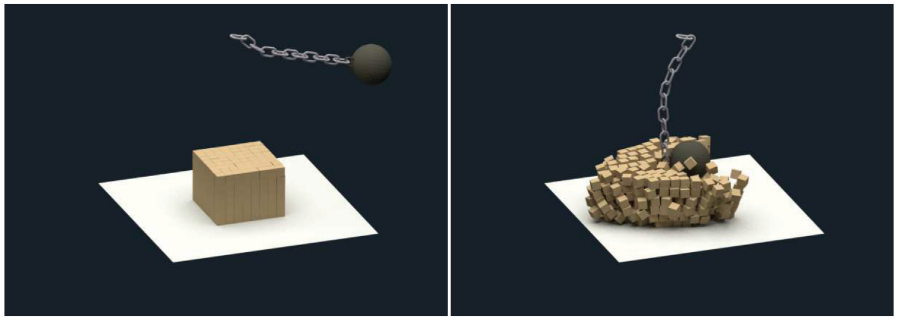
Oprócz zastosowań naukowych wykorzystywane są również w renderingu oraz animacji. Czas renderu kilkusekundowych scen filmu animowanego, w wysokiej jakości, może być liczony w dniach, a nawet tygodniach. Na przykład film *Potwory kontra Obcy* renderowany była na specjalnej farmie obliczeniowej łącznie przez 40 milionów roboczogodzin. Jedna klatka w rozdzielczości kinowej z udziałem renderowanego robota Devastatora w filmie *Transformers – Zemsta Upadłych* liczyła się na specjalnej farmie renderującej 72 godziny. Sam robot zbudowany był z 52 632 elementów, 11 716 127 krzywych oraz 6 467 tekstur [3]. Ze względu na czas jaki należałoby poświęcić na wyznaczenie rozwiązania generowania animacji na komputerze domowym jest w większości przypadków niemożliwe.

2. Projekt oraz implementacja rozproszonego systemu obliczeniowego dedykowanego wizualizacjom 3D

Zaproponowane w niniejszym artykule rozwiązanie ma cechować się uniwersalnością, skalowalnością, niskim kosztem wdrożenia i utrzymania. Celem obniżenia kosztów, autorzy zdecydowali się na wykorzystanie darmowego oprogramowania opartego na licencji GNU GPL. Do utworzenia animacji testowej, wykorzystywanej później w eksperymencie, wybrano program Blender [4]. Jest oprogramowanie, które posiada niemalże wszystkie funkcje dostępne w komercyjnych rozwiązaniach m.in. Autodesk Maya (3 675$) lub Autodesk 3DS (1 470$). Środowisko pracy Linux Mint było emulowane w programie VirtualBox dzięki czemu niezależnie od warstwy hardware'owej i software'owej komputera, zawsze będzie pracowało w taki sam sposób. Połączenie w jedną sieć globalną realizowane będzie z wykorzystaniem pakietu OpenVPN.

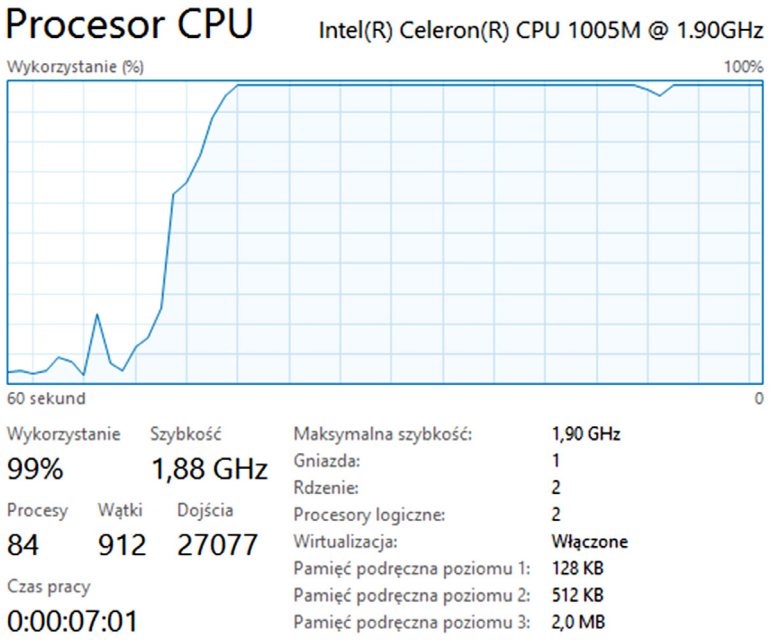
2.1. Przygotowanie animacji

W programie Blender przygotowano 100-klatkową animację, na której kula zamocowana na łańcuchu pod wpływem siły ciężkości uderza w prostopadłościan ułożony z 384 sześcianów [5]. Wszystkim zamodelowanym elementom układu nadano właściwości *Rigid Body* (pl. bryła sztywna). Dzięki takiemu zabiegowi ogniwa łańcucha nie będą się przenikać, a ruch i-tego elementu jest powiązany z ruchem $i - 1$. Górnemu ogniwu odebrano wszystkie stopnie swobody translacyjne i rotacyjne, a w jego geometrycznym środku ciężkości znajduje się środek obrotu całego układu. Elementami kolizyjnymi na torze ruchu wahadła fizycznego są 384 sześciany ułożone w prostopadłościan (Rys. 2).



Rys. 2. Widok układu w klatce 10 oraz 47

Animacja została obliczona na komputerze Lenoro B590 wyposażonym w dwurdzeniowy procesor Intel Celeron 1005M taktowanym zegarem 1.90 Hz. Render animacji o rozdzielczości 1024×720 oraz próbkowaniu 75 trwał 1 godzinę 50 minut i 28 sekund. W trakcie trwania obliczeń procesor wykorzystany był niemalże w 100% przez cały czas trwania obliczeń, uniemożliwiając tym samym płynne wykonywanie innych procesów (Rys. 3). Wynik tego testu będzie punktem odniesienia dla eksperymentu w systemie rozproszonym.



Rys. 3. Wykorzystanie dwurdzeniowego procesora w trakcie obliczeń animacji

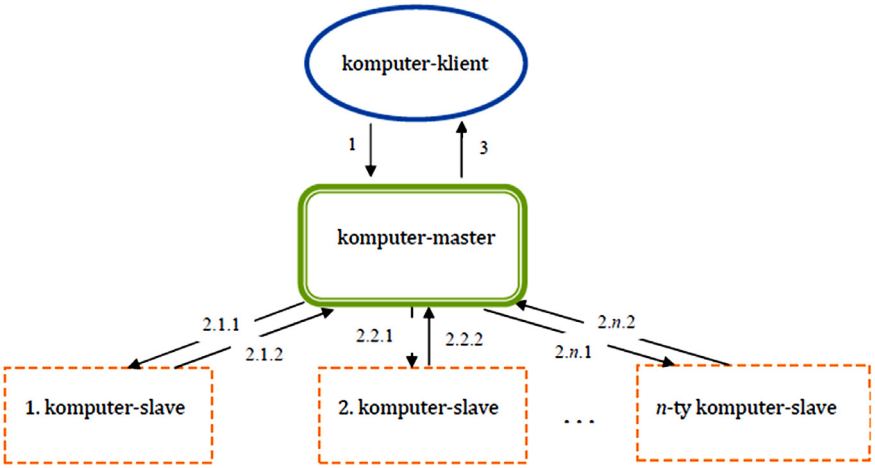
2.2. Procedura przepływu informacji w systemie

Istnieje wiele farm renderujących, których cena zależy bezpośrednio od ilości procesorów (lub innych jednostek liczących) albo czasu obliczeń (np. *Pixek Plow* 0.005 USD za GHzh[1], *Render4you* 0.016 € za GHzh) [7].

Główną wadą korzystania z tego rozwiązania jest przede wszystkim brak informacji o sposobie przetwarzania renderu, gdzie dane tymczasowe są zapisane oraz w jaki sposób są zabezpieczone przed dostępem osób trzecich.

Przed przystąpieniem do prac projektowych opracowano procedurę przepływu danych. System zbudowany jest z trzech podstawowych typów komputerów: komputera-klienta, komputera typu *master* oraz komputera typu *slave* [8]. Procedura rozpoczyna się przesłaniem zadania obliczeniowego z komputera-klienta do komputera typu *master*, który następnie dzieli plik na pojedyncze klatki i przesyła je kolejno do poszczególnych komputerów typu *slave*. Po zakończeniu obliczeń komputery typu *slave* przesyłają rozwiązanie do komputera typu master, który składa je w całość i wysyła do komputera-klienta (Rys. 4).

[1] GHzh – jednostka czasu renderu, według której obliczana jest cena obliczeń: 1 GHzh = 1 GHz (CPU) · 1 h



Rys. 4. Procedura przepływu informacji w systemie

Założony przepływ informacji może być zrealizowany w Blenderze z wykorzystaniem modułu Network Render. Jest to dodatek, który ułatwia przeprowadzanie obliczeń rozproszonych przez przydzielenie typu komputera (*klient – master – slave*) do odpowiedniej instancji Blendera. Po włączeniu, Network Render pojawia się w rozwijanej liście silników obliczeniowych. Ważne jest to, aby pamiętać, że Network Render pracuje tylko w sieci lokalnej.

Przed rozpoczęciem pracy należy upewnić się, że wszystkie komputery, na których planowane są obliczenia znajdują się w jednej sieci lokalnej. W pierwszym kroku należy skonfigurować komputer typu *master* (serwer) podając jego numer IP oraz numer portu komunikacyjnego (domyślnie: 8000). Następnie należy ustawić parametry wszystkich instancji typu *slave* oraz *klient.* Podobnie jak wcześniej należy podać numer IP mastera oraz ten sam numer portu, który wpisano w parametrach *mastera*.

2.3. Projekt i implementacja systemu

Network Render jest narzędziem o szerokim spektrum zastosowań z możliwością implementacji w profesjonalnych oraz amatorskich farmach renderowych. Autorzy artykułu podjęli próbę zaprojektowania, a następnie implementacji rozproszonego systemu obliczeniowego do renderingu 3D wykorzystującego moduł Network Render.

W założeniach projektowany system miałby być uniwersalny (tj. pracować niezależnie od warstwy hardware'owej i software'owej), skalowany (im więcej komputerów pracuje nad zadaniem obliczeniowym, tym wydajność i efektywność systemu jest wyższa), autonomiczny (niewymagający ingerencji użytkownika) oraz łatwy w implementacji (Rys. 5).

Rys. 5. Założenia projektowe systemu rozproszonego

Aby system pracował zawsze tak samo, niezależnie od warstwy sprzętowej oraz systemu operacyjnego, autorzy pracy zdecydowali się na pracę w wirtualnych maszynach przygotowanych w VirtualBoxie [9]. Porównując wady i zalety emulowanych systemów, tj. Linuxa i Windowsa, ostatecznie zdecydowano się na Linuxa w wersji Mint XFCE 32-bit. Jest to darmowy system operacyjny o niskich wymaganiach sprzętowych. Cechuje go bardzo dobre zarządzanie czasem oraz łatwa konfiguracja interfejsu sieciowego. Poprawnie skonfigurowany Linux jest odporny na nieautoryzowane działania osób nieuprawnionych. Do pracy systemu nie jest wymagane środowisko graficzne, a obsługa z konsoli jest przyjazna użytkownikowi.

Utworzone zostały trzy typy wirtualnych maszyn: master, slave oraz klient, na których zostały skonfigurowane odpowiadające im instancje Blendera wraz z modułem Network Render.

2.4. Konfiguracja Open VPN

Na obecnym etapie system może pracować w sieci lokalnej. Można sklonować instancję wirtualnej maszyny typu *slave* czyniąc system bardziej przystosowanym do potrzeb użytkownika. Jednakże, autorom pracy zależało na tym, aby system działał na jak największej liczbie komputerów pracujących w sieci globalnej. Jednym z rozwiązań tego problemu jest wykorzystanie pakietu OpenVPN [10], którego zadaniem będzie tworzenie tuneli odizolowanych od sieci publicznej. Dzięki temu połączenie VPN zapewnia wysoki standard bezpieczeństwa.

Bezpieczeństwo połączenia zapewniane przez OpenVPN wynika z uwierzytelnienia, autoryzacji oraz szyfrowania połączenia. Podczas konfiguracji połączenia generowane są certyfikaty, klucze prywatne oraz publiczne, dzięki którym zarówno strona serwera jak i klienta przeprowadza proces uwierzytelnienie oraz autoryzacji [11].

Kroki tworzenia połączenia OpenVPN opisane w dalszej części artykułu opierają się na wykorzystaniu i uruchomieniu skryptów napisanych w BASH dostępnych w pakiecie *easy-rsa*.

Ustanawianie połczenia OpenVPN zaleca się rozpocząć od utworzenia własnego centrum certyfikującego. Aby podnieść bezpieczeństwo połączenia, można utworzyć tajny klucz z wykorzystaniem algorytmu Diffie-Hellmana [12]. Algorytm ten wykorzystuje arytmetykę multiplikatywnych grup modulo p, gdzie p jest możliwie dużą liczbą pierwszą, a g pierwiastkiem pierwotnym modulo p. Następnie należy wygenerować klucz prywatny i publiczny dla serwera wywołując z linii poleceń skrypt *shbuild-key-server*. Kolejnym krokiem jest utworzenie klucza prywatnego i publicznego dla klienta połączenia w oparciu o skrypt *build-key*. Pliki konfiguracyjne serwera oraz klienta należy uzupełnić podając nazwy certyfikatu, klucza prywatnego i publicznego. Dodatkowo w pliku konfiguracyjnym serwera należy zwrócić uwagę na adres IP oraz numer nasłuchiwanego portu.

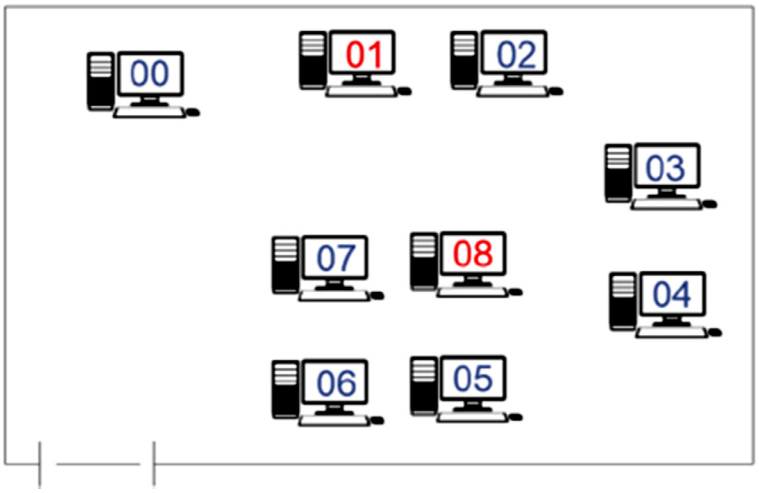
Właściwe jest, aby dla każdego komputera łączącego się z serwerem OpenVPN wygenerować inny, unikalny klucz prywatny oraz publiczny. W tym przypadku autorzy pracy w pliku konfiguracyjnym serwera zezwolili na duplikowanie kluczy. Oznacza to, że wirtualne maszyny mogą być dowolnie klonowane, w zależności od indywidualnych potrzeb, bez potrzeby generowania nowych kluczy będą się automatycznie łączyć z serwerem.

Z wykorzystaniem systemowego polecenia *rc-update.d* dodano skrypt uruchomieniowy napisany w języku BASH. Skrypt odpowiedzialny jest za uruchomienie Blendera z odpowiedniego pliku, zawierającego parametry oraz rodzaj instancji Network Render. Dodatkowo, jeżeli nie zostanie nawiązane połączenie OpenVPN, co 60 sekund komputer będzie odpytywał serwer i podejmował próbę nawiązania połączenia. Z wykorzystaniem reguł *iptables* ruch jest przekierowany ruch z interfejsu *tap0* na *eth0* i odwrotnie co zapewnia łączność wzajemną łączność pomiędzy wszystkimi komputerami w sieci VPN. Zaletą dodania tego skryptu uruchomieniowego jest to, że Blender oraz OpenVPN uruchamiają się wraz ze startem systemu. Tym samym, użytkownik nie musi posiadać hasła do systemu, aby manualnie połączyć się z serwerem i rozpocząć pracę.

Zaprojektowany system spełnia wszystkie narzucone wymagania. Rola użytkownika udostępniającego moc obliczeniową sprowadza się do pobrania wirtualnej maszyny i uruchomienia jej w dowolnym programie wirtualizacyjnym. Od tego momentu, jego moc obliczeniowa jest współdzielona użytkownikom systemu.

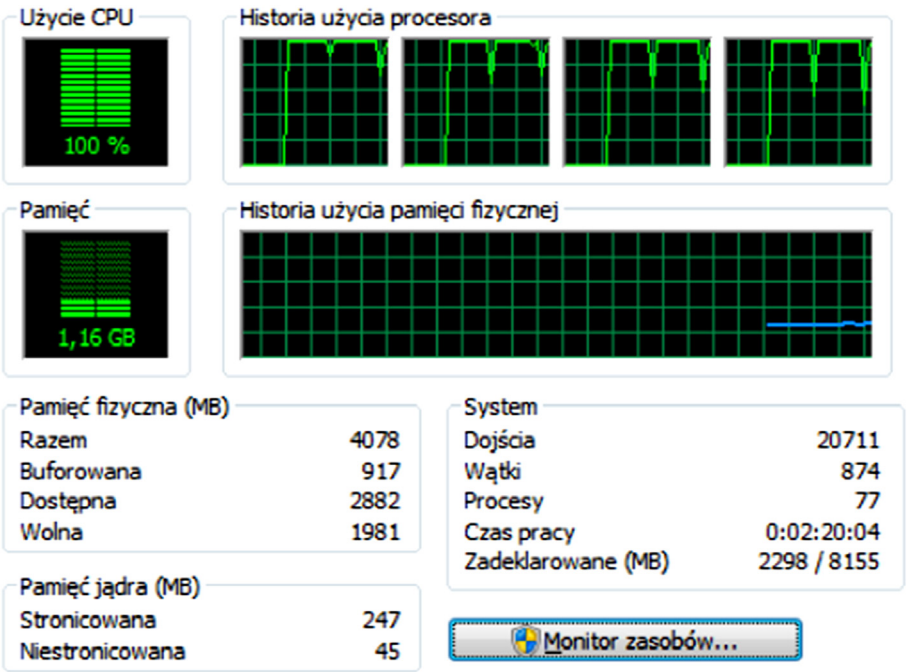
3. Testy wydajności systemu

System został zaimplementowany w sali 313 w budynku D-1 Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie. Eksperyment przeprowadzono na 7 stanowiskach komputerach (Rys. 6). Komputery cechuje homogeniczna budowa warstwy software'owej i hardware'owej: wyposażone są w czterordzeniowe procesory Intel Core i5 taktowane zegarem 3.1GHz oraz pracują na systemie operacyjnym Windows 7.



Rys. 6. Schemat sali komputerowej 313 (stanowiska 01 oraz 08 nie brały udziału w testach)

Pierwszy test przeprowadzono na stanowisku 00 bez wykorzystania systemu rozproszonego. Obliczenia przeprowadzono w czasie 19 minut i 38 sekund. Niemniej jednak, w trakcie trwania obliczeń wszystkie rdzenie procesora były zajęte przez Blendera (Rys. 7), uniemożliwiając tym samym płynne wykonywanie innych procesów.

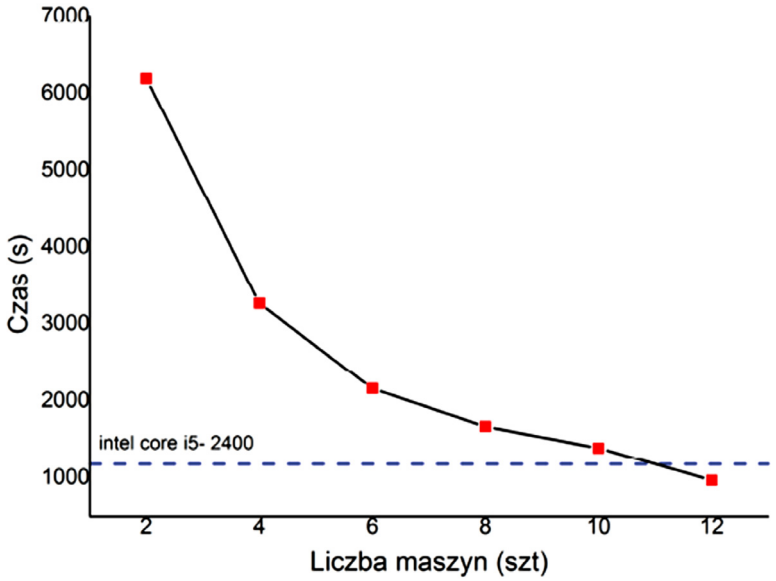


Rys. 7. Wykorzystanie procesora w trakcie obliczeń na stanowisku 00

Następnie, na każdym stanowisku komputerowym zamontowano odpowiednią wirtualną maszynę:

- stanowisko 00: Blender Master oraz Serwer OpenVPN,
- pozostałe stanowiska: Blender Slave oraz OpenVPN Klient.

W celu sprawdzenia działania oraz wydajności systemu przeprowadzono testy na 2,4,6,8,10 oraz 12 wirtualnych maszynach (Rys. 8).



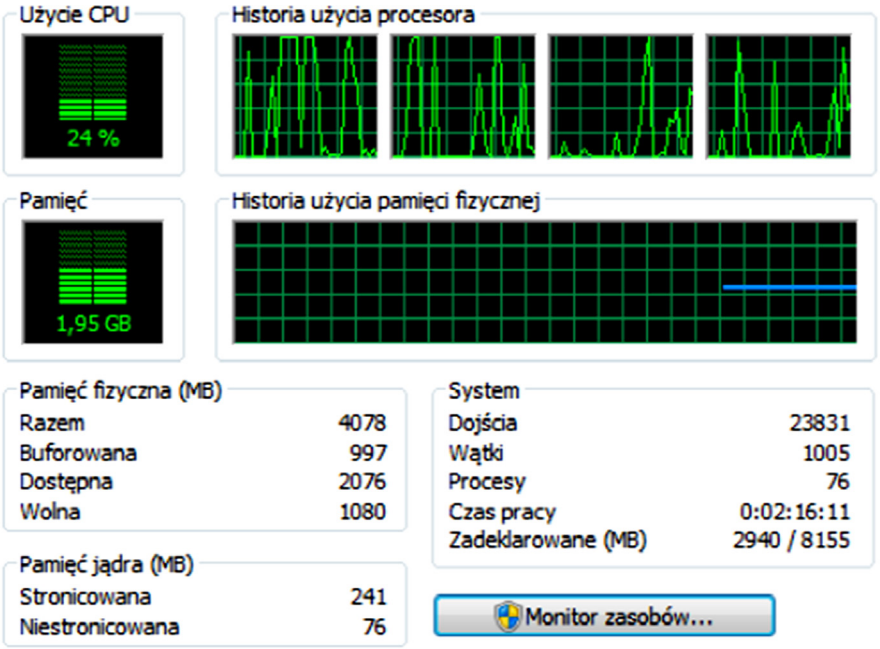
Rys. 8. Zależność czasu obliczeń od ilości maszyn oliczeniowych

W programie MATLAB dopasowano punkty pomiarowe do przebiegu krzywej potęgowej. Krzywa dana jest zależnością

$$f(x) = 12070\,x^{-0.9584}$$

gdzie x to liczba komputerów typu *slave* pracujących w systemie. Zależność ta pozwala wyznaczyć czas obliczeń dla znanej liczby x. Obowiązuje dla systemu obliczeniowego zbudowanego z takiej samej warstwy hardware'owej, jak system testowy.

Zauważono, że w trakcie obliczeń w systemie rozproszonym obciążenie procesora zależy od ilości uruchomionych wirtualnych maszyn. Jedna wirtualna maszyna wykorzystuje jeden rdzeń procesora (Rys. 9). W tym przypadku, jedna instancja wykorzystuje 25% mocy obliczeniowej procesora.



Rys. 9. Wykorzystanie mocy obliczeniowej w przypadku obliczeń prowadzonych w systemie rozproszonym

4. Wnioski

W artykule zaprezentowana została możliwość wykorzystania obliczeń rozproszonych w zastosowaniach renderingu i animacji. Zaproponowane rozwiązanie stanowi alternatywę dla płatnych, zamkniętych farm obliczeniowych.

Wykazana została zwiększona wydajność obliczeń w systemie. Zaletą zaproponowanego rozwiązania jest niskie wykorzystanie mocy obliczeniowej przy jednoczesnym wzroście wydajności obliczeń. Przyjęte założenia zostały spełnione. Zaproponowane rozwiązanie cechuje uniwersalizm, może być zaimplementowane na każdym komputerze niezależnie od głównego systemu operacyjnego. Użytkownik, który chce udostępnić swoją moc obliczeniową musi tylko pobrać i uruchomić wirtualną maszynę utworzoną przez autorów pracy, a następnie uruchomić ją w dowolnym programie wirtualizacyjnym. Działanie systemu nie wymaga od użytkownika logowania się do systemu. Cały proces został zautomatyzowany: ustanowienie połączenia OpenVPN, uruchomienie Blendera oraz dodatku Network Render, a następnie wczytanie pliku z ustawieniami Blendera.

Literatura

[1] Główny Urząd Statystyczny, *Społeczeństwo informacyjne w Polsce w 2014* r. [online], [dostęp: 04 czerwca 2015], Dostępny w Internecie: < http://stat.gov.pl/obszarytematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwoinformacyjne/spoleczenstwo-informacyjne-w-polsce-w-2014-r-,2,4.html>

[2] *Programowanie równoległe i rozproszone*, praca zbiorowa pod red. A. Karbowskiego i E. Niewiadomskiej-Szynkiewicz, Warszawa, Oficyna Wydawnicza Politechniki Warszawskiej, 2009.

[3] *How To: Building Your Own Render Farm* [online], [dostępny: 2 czerwca 2015], Dostępny w Internecie: < http://www.tomshardware.com/reviews/render-farmnode,2340.html>

[4] Chlipalski P., *Blender 2.69 - Architektura i Projektowanie*, Gliwice, Wydawnictwo Helion, 2014.

[5] *Introduction to Rigid Body Simulations* [online], [dostępny: 25 kwietnia 2015], Dostępny w Internecie: < http://www.blenderguru.com/tutorials/quick-tutorial-makea-wrecking-ball-with-rigid-body-physics/>

[6] *Complete katalog of all online render-farm services* [onlines], [dostępny: 06 czerwca 2015], Dostępny w Internecie: <http://rentrender.com/all-render-farms-list/>

[8] Ahmar Abbas, *Grid computing: a practical guide to technology and applications*, Massachusetts, Charles River Media Inc., 2004.

[9] Serafin M., *Wirtualizacja w praktyce*, Gliwice, Wydawnictwo Helion, 2011.

[10] *Praktyczna implementacja sieci VPN na przykładzie OpenVPN* [online], [dostęp: 03 maja 2015], Dostępny w Internecie: < http://sekurak.pl/praktyczna-implementacjasieci-vpn-na-przykladzie-openvpn/>

[11] *Get OpenVPN up and running, enjoy your privacy* [online], [dostęp: 01 maja 2015], Dostępny w Internecie: < http://parabing.com/2014/06/openvpn-on-ubuntu/>

[12] Schneier B., *Kryptografia dla praktyków: protokoły, algorytmy i programy źródłowe w języku C*. Warszawa, Wydawnictwa Naukowo-Techniczne, 2002, s. 629-631.

The application of distributed computing system in 3D visualizations

Abstract

In this paper the methodology of design and implementation of distributed computing system based on an example of 3D visualization is introduced. The proposed system is supposed to be universal, scalable and user-friendly. The system depends on users who free of charge share their computing power. The solution has been designed to rely on free license software and costless operating system Linux. The proposed in the article solution is a great alternative of closed, commercial computing systems.

Keywords: rendering farms, distributed computing, rendering, animation