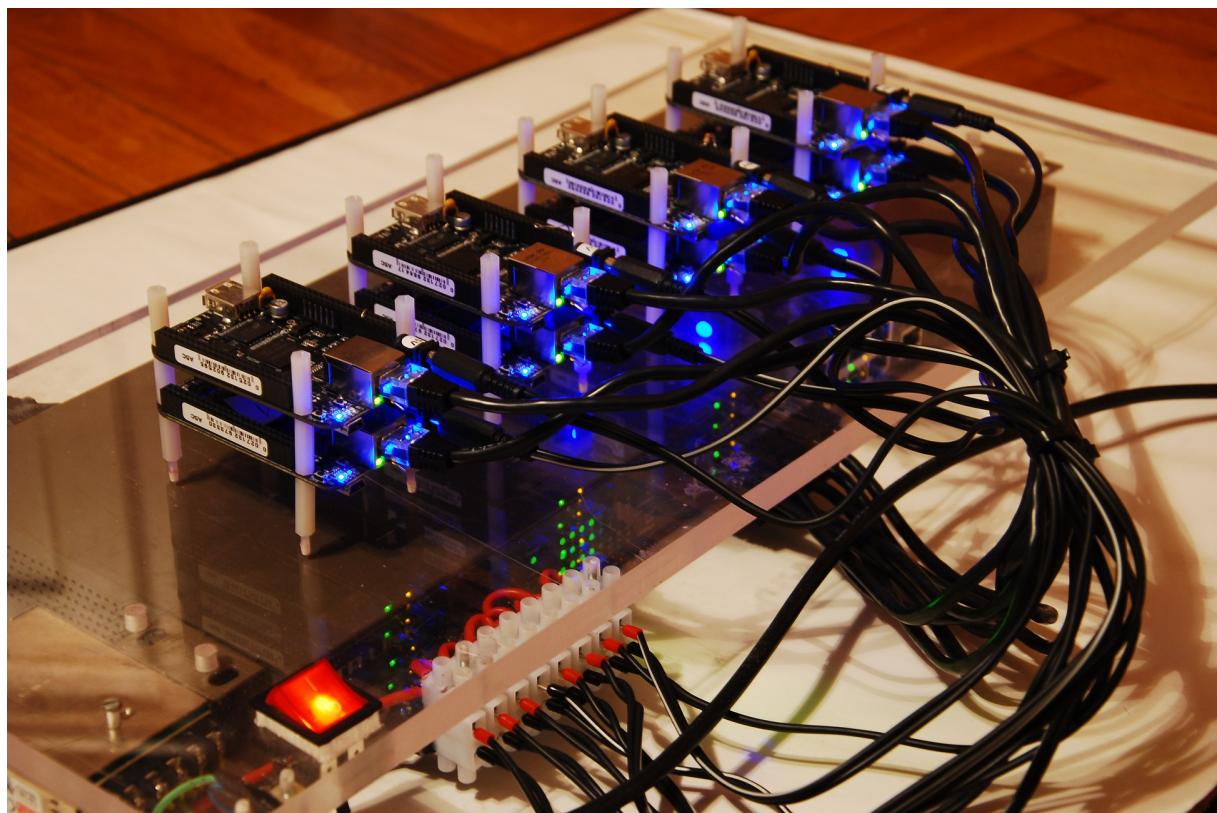


Klaster do obliczeń równoległych  
Centrum Fizyki Teoretycznej PAN  
v1.2

Piotr Zdunek  
p.zdunek@stud.elka.pw.edu.pl

18.12.2013



## **Spis treści**

# 1 Wstęp

Poniższy dokument jest poradnikiem wykorzystania klastra obliczeniowego zbudowanego w Centrum Fizyki Teoretycznej PAN na potrzeby testowania oprogramowania równoległego napisanego przy wykorzystaniu biblioteki OpenMPI. Każdy może skorzystać z klastra i rozszerzyć swoją wiedzę z zakresu działania systemów rozproszonych, systemu operacyjnego Linux, programowania w C/C++ lub Fortranie oraz poznać działanie komputera Beaglebone Black z których został zbudowany klaster.

Poradnik jest skierowany dla każdego: do uczniów, studentów, doktorantów pragnących z szlifować swoje pasje związane z komputerami i programowaniem równoległym.

**Struktura poradnika** Poradnik jest podzielony na 4 główne rozdziały:

- Struktura i zasada działania klastra
- Dostęp do klastra i zasady pisania projektów
- Programowanie równolegle w OpenMPI
- Profilowanie programów równoległych - dla zaawansowanych

Programowania na klastrze jest stosunkowo proste i można się bardzo wiele nauczyć przy pracy z tym urządzeniem. Życzę miłego korzystania z klastra.

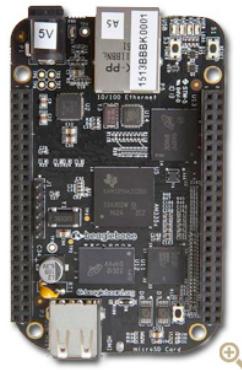
**Podobne projekty** Wiele wiodących ośrodków naukowych wykonało podobne projekty - głównie oparte o Raspberry Pi [?]. [?] [?].

## 2 Struktura i zasada działania klastra

W tym rozdziale zajmiemy się podstawami działania klastrów obliczeniowych oraz przedstawiona zostanie struktura naszego *superkomputera*.

**Klaster** jest systemem połączonych ze sobą komputerów w celu współdzielenia obliczeń. Taki system można zbudować na bardzo wiele sposobów i dzisiejsze wielkie portale jak np. *facebook* nie byłyby w stanie obsłużyć tak dużej ilości użytkowników gdyby nie klastry - inaczej superkomputery.

**Beaglebone Black** Nasz klaster zbudowany jest z 8 komputerów Beaglebone Black [?], połączonych ze sobą poprzez sieć Ethernet (jak komputery w domu). Są to tanie komputery dla entuzjastów pragnących poznać tajniki elektroniki, mechatroniki, budowania systemów sterowania; dzięki zastosowaniu Wolnego i Otwartego Oprogramowania (system Linux) możliwe jest elastyczne wykorzystywanie mocy takich komputerów.



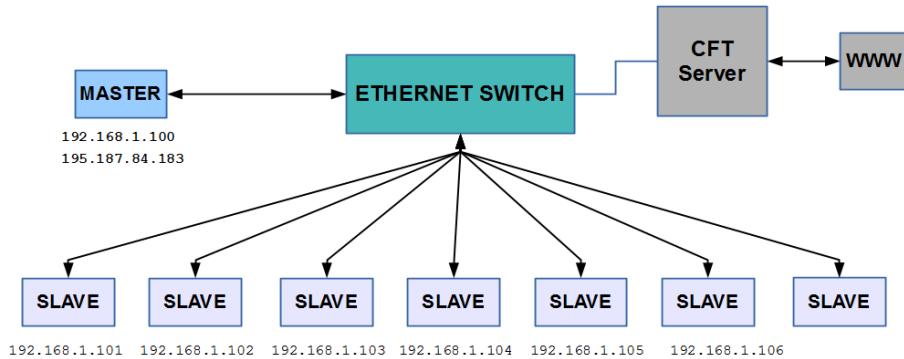
Rysunek 1: Tak wygląda komputer Beaglebone Black - rozmiaru karty kredytowej

Jak do każdego komputera, do BBB (**BeagleBone Black**) możemy podłączyć klawiaturę i myszkę i normalnie korzystać poprzez interfejs graficzny lub konsolę. Moc obliczeniowa jest porównywalna z mocą typowego nowoczesnego telefonu komórkowego albo komputera sprzed 8-10 lat. Na BBB jest zainstalowany system operacyjny

Linux. To Wolne i Otwarte Oprogramowanie, zatem jest w pełni legalne i darmowe dla każdego użytkownika. Jak każdy system, wymaga znajomości podstaw jego obsługi, więc tu wymagana jest podstawowa znajomość polecień konsoli systemu Linux.

## 2.1 Struktura klastra

Poniższy diagram przedstawia strukturę klastra; każdy z komputerów ma przypisany statyczny adres IP i dzięki wykorzystaniu specjalnej biblioteki OpenMPI istnieje możliwość rozdzielenia obliczeń na komputery występujące w tej samej podsieci, jeśli zawierają takie samo oprogramowanie oraz binarny program który chcemy uruchomić.



Rysunek 2: Struktura klastra

W uproszczeniu jest to po prostu 8 komputerów połączonych ze sobą w sieć. Jeden jest komputerem głównym z którego są wykonywane programy i który wydaje rozkazy innym komputerom. Istnieje wiele typów klastra, różniących głównie ze względu na sposób komunikacji lub architekturę jednostek obliczeniowych, zbudowany klaster jest typu Beowulf [?].



Rysunek 3: Beaglebone Cluster

## 2.2 Konfiguracja systemu

Na każdym z komputerów znajduje się ten sam użytkownik o nazwie *cluster*, który uruchamia programy na swoim węźle. Warunkiem uruchomienia programu równolegle na klastrze jest to aby ścieżka do pliku programu była taka sama na każdym komputerze. Dlatego też na każdym węźle jest ten sam użytkownik a programy

znajdują się w katalogu `/home/cluster/software` który jest współdzielony między wszystkie komputery - jest to serwer NFS.

Dla jasności komunikacji pomiędzy węzłami w każdym węźle klastra dodane zostały adresy komputerów do pliku `hosts`. Komputery w sieci klastra można identyfikować za pomocą jasnych nazw, a nie adresów IP.

```
127.0.0.1      localhost
192.168.1.2    theta1

192.168.1.100  master
192.168.1.101  slave1
192.168.1.102  slave2
192.168.1.103  slave3
192.168.1.104  slave4
192.168.1.105  slave5
192.168.1.106  slave6
192.168.1.107  slave7
```

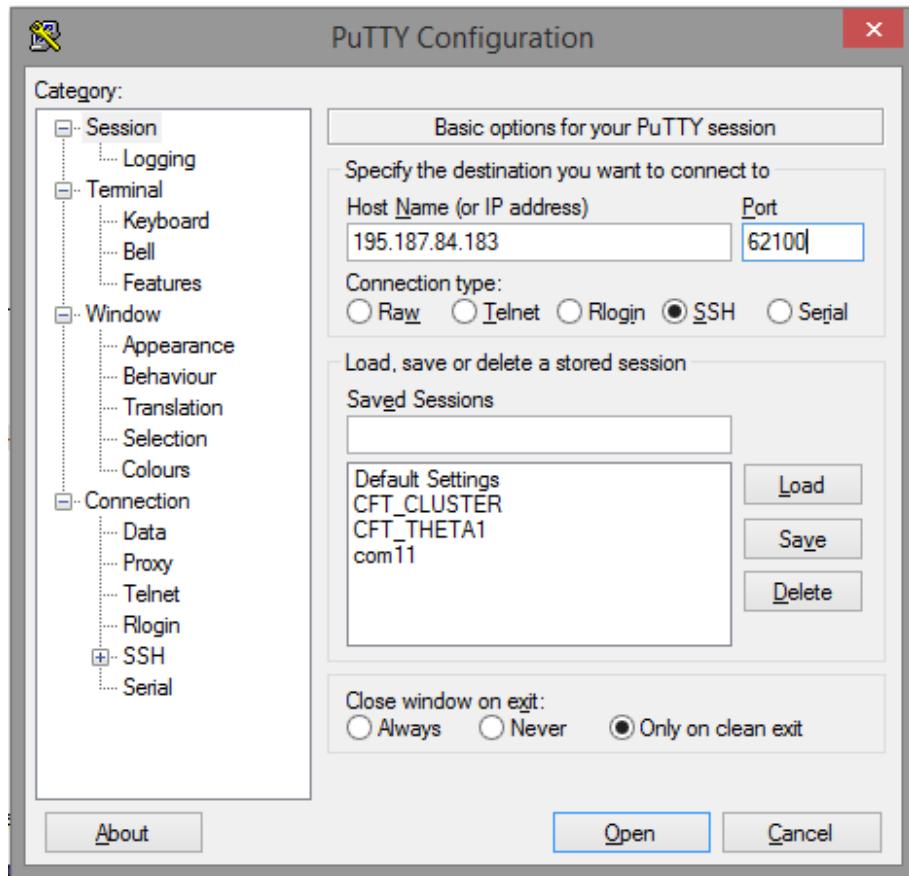
Rysunek 4: Zawartość pliku `/etc/hosts`

### 3 Dostęp do klastra i zasady pisania projektów

Z klastrem łączymy się za pomocą protokołu SSH (*Secure Shell*), jest to szyfrowany protokół komunikacyjny w sieciach komputerowych. Można to zrobić z (prawie) dowolnego komputera. Połączenie najpierw jest zestawiane z serwerem Centrum Fizyki Teoretycznej PAN - Theta1, do powłoki [?] o ograniczonym dostępie. Następnie po wydaniu odpowiedniej komendy można połączyć się z klastrem. Poniżej znajduje się opis zasad dostępu do klastra oraz wytlumaczenie krok po kroku w jaki sposób można się z nim połączyć.

#### 3.1 Dostęp z systemu Windows

Aby uzyskać dostęp do klastra z poziomu systemu Windows potrzebujemy programu PuTTY [?]. Konfigurujemy program jak na poniższym obrazie:



Rysunek 5: Konfiguracja PuTTY

Wpisujemy nazwę użytkownika i hasło.

```
login:cluster
pass:cluster
```

Voila! Jesteśmy już zalogowani na klastrze i możemy zacząć pisać programy równoległe w OpenMPI. Jednak najpierw wypada poznać podstawy pracy z systemem operacyjnym Linux. Polecam podane w przypisach strony internetowe/artykuły/filmy [?][?][?].

### 3.2 Dostęp z systemu Linux

Dostęp do klastra w systemie Linux jest bardzo podobny. Uruchamiamy konsolę i wydajemy komendę  
`ssh -l cluster 195.187.84.183 -p 62100 -X -Y`  
następnie podajemy hasło do klastra.

```
pass:cluster
```

Przy pierwszym uruchomieniu należy zezwolić na dodanie kluczy RSA.

### 3.3 Zdalny dostęp do powłoki graficznej

Istnieje możliwość połączenia się zdalnie z klastrzem poprzez powłokę graficzną.

#### Linux

- Uruchamiamy dwa terminale tekstowe
- Na pierwszym uruchamiamy drugi X serwer wydając komendę  
`X :1 -r`  
komputer przełączy się na tę pustą powłokę, aby powrócić do pierwszego serwera X należy nacisnąć kombinację klawiszy *CTRL+ALT+F7*

- Przechodzimy na drugi terminal i dodajemy zmienną środowiskową  

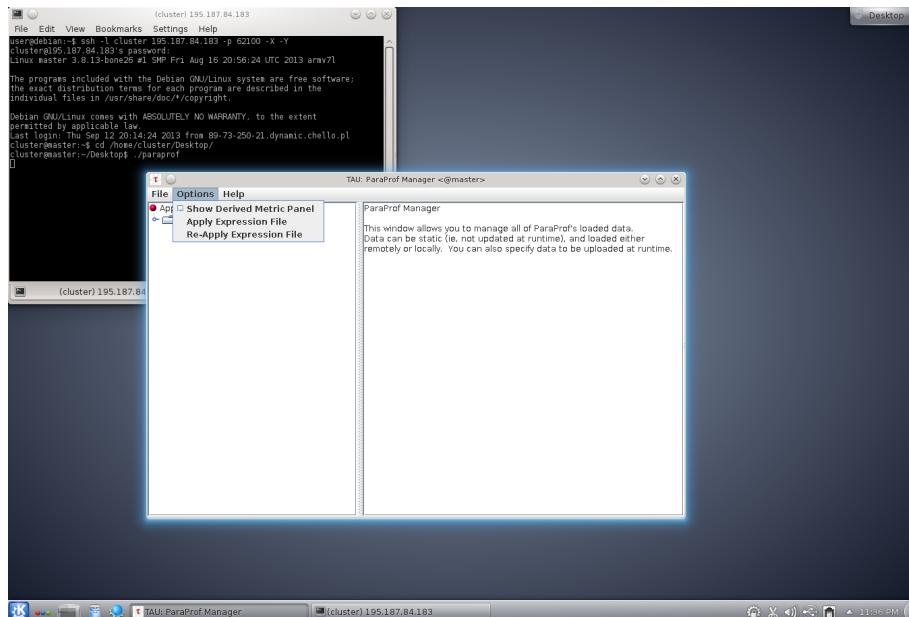
```
export DISPLAY=:1
```
- Logujemy się na klastrem za pomocą komendy  

```
ssh -l cluster 195.187.84.183 -p 62100 -X -Y
```
- Wywołujemy uruchomienie sesji  

```
/etc/X11/Xsession
```

  
 i czekamy aż pojawi się obraz

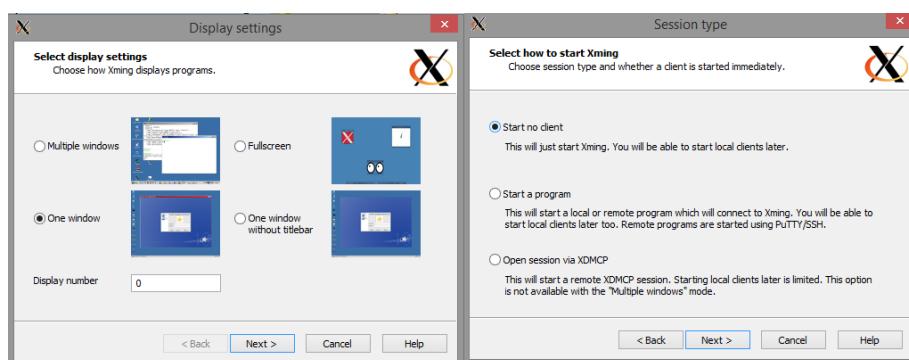
Najlepszym sposobem na wyjście z sesji X klastra jest zakończenie procesu Xserwera poprzez kombinację klawiszy **CTRL+ALT+BACKSPACE** - **uwaga** nie działa na wszystkich współczesnych dystrybucjach. Można również uruchamiać same aplikacje, bezpośrednio z poziomu konsoli.



Rysunek 6: Paraprof uruchomiony bezpośrednio z konsoli

**Windows** Do połączenia z klastrem za pomocą powłoki graficznej potrzebny jest darmowy program Xming do pobrania ze strony [?].

- Uruchamiamy program Xming, z ustawieniami jak na poniższym obrazku.

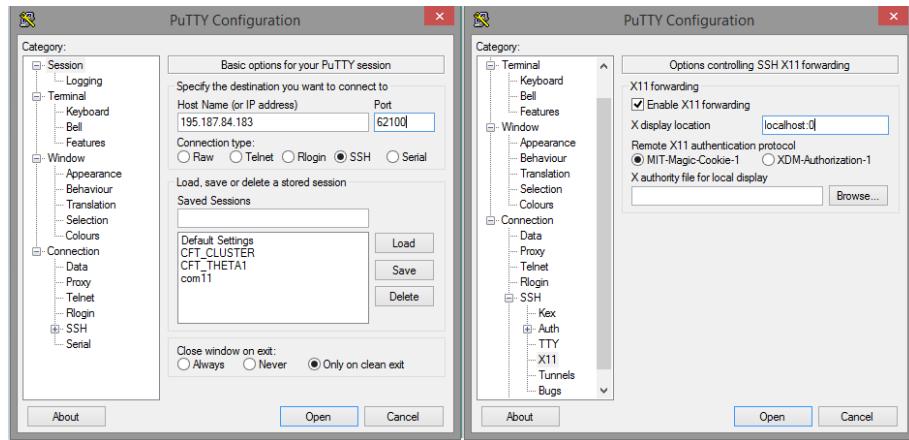


Rysunek 7: Ustawienia programu Xming

Po uruchomieniu pojawi się puste okno serwera X11.

- Łączymy się z klastrem za pomocą programu PuTTY z odpowiednimi ustawieniami X11.

- Po zalogowaniu na klastr wydajemy komendę `/etc/X11/Xsesion` i po chwili powinniśmy zobaczyć pulpitu klastra.



Rysunek 8: Konfiguracja serwera X11 w programie PuTTY

### 3.4 Zasady korzystania z klastra

Dostęp do klastra jest ograniczony tj. tylko jedna osoba/grupa osób otrzyma hasło do konta *cluster* na serwerze CFT, ze względu na ograniczenia mocowe klastra. Hasło można uzyskać wysyłając email na adres [ctpcluster@gmail.com](mailto:ctpcluster@gmail.com) z odpowiednimi informacjami tj.

- imię i nazwisko, kontakt do osoby odpowiedzialnej za projekt
- nazwę projektu jaki chce się wykonać na klastrze
- czas potrzebny do wykonania projektu

Pod adresem <http://goo.gl/lmt21X> dostępny jest kalendarz dostępu do klastra. Na koniec po wykonaniu projektu będzie należał złożyć krótki raport z wykorzystania klastra.

### 3.5 Sprawozdanie z projektu

Każdy wykonany projekt należy krótko opisać. W katalogu głównym projektu należy umieścić dokument, najlepiej w formacie odt, doc lub pdf, w którym muszą być zawarte następujące informacje:

- Nazwa projektu
- Skład zespołu
- Opis algorytmu
- Uzyskane przyspieszenie obliczeń
- Porównanie wydajności - np. w formie czasu obliczeń w porównaniu do komputera klasy PC

Przykład sprawozdania:

# Klaster obliczeniowy CFT PAN

## Sprawozdanie z projektu

Nazwa projektu: Mnożenie macierzy

### 1. Skład osobowy zespołu:

Piotr Zdunek

### 2. Opis algorytmu:

N – wielkość macierzy

M – ilość procesów (węzłów klastra)

- Generuj dwie losowe tablice o wielkości N x N
- Rozdziel po równo zakresy iteracji pętli każdemu procesowi
  - Warunek na  $N \rightarrow N/M \bmod 4 = 0$
- Wykonaj mnożenie macierzy za pomocą 3 pętli for
- Zsumuj wyniki obliczeń do jednej zmiennej za pomocą funkcji `MPI_Reduce()`.

Dodatkowo w programie znajdują się tzw. `milestone's` pomiędzy którymi mierzony jest czas pracy programu.

Uruchomienie programu:

`mpirun -np 8 --hostfile hostfile ./matmul.exe -n 1024`

### 3. Przyspieszenie obliczeń dzięki OpenMPI

Dla M = 1, N=1024 czas obliczeń wynosi średnio 85 s.

Dla M=8, N=1024 czas obliczeń wynosi średnio 11 s.

Przyspieszenie obliczeń:  $85/11 = 7.27$

### 4. Porównanie wydajności do komputera klasy PC

Procesor Intel Core i5-3447K 4 rdzenie, program jednowątkowy.

M=1, N=1024 czas obliczeń 13.37 s

### 5. Podsumowanie

Bezpośredni algorytm mnożenia macierzy daje możliwość bardzo dużego przyspieszenia obliczeń dzięki OpenMPI. Co ciekawe klaszter liczy wynik szybciej niż komputer klasy PC, który pobiera znacznie więcej energii, jednak należałoby przeprowadzić więcej testów aby dokładnie sprawdzić dla jakich danych wejściowych który komputer jest szybszy.

Kod programu jest przykładem z poradnika „MPI in Thirty Minutes” źródło:

<http://www.linux-mag.com/id/5759/>

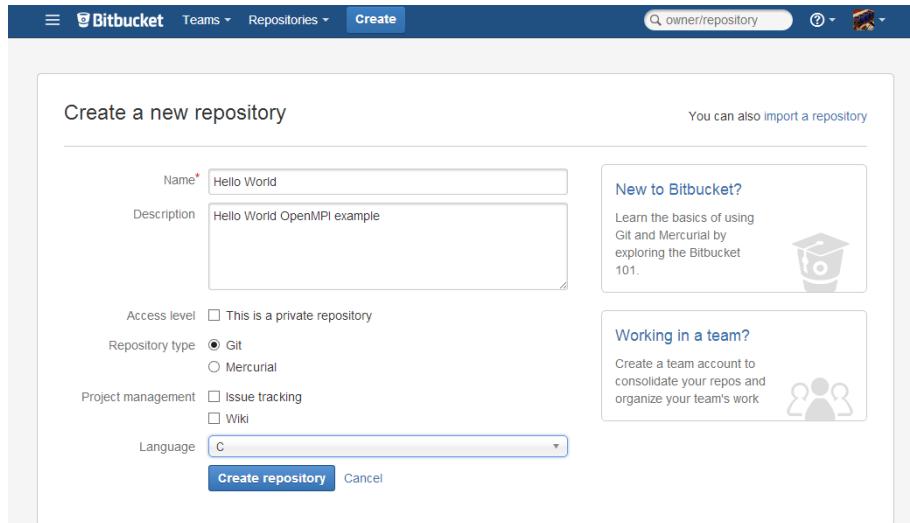
Rysunek 9: Sprawozdanie z projektu

Przykładowy format sprawozdania można pobrać z adresu:<http://goo.gl/1D9ZVc>.

### 3.6 Repozytorium GIT

Projekt klastra jest ogólnodostępny tak jak i projekty. Autorzy proszeni są o umieszczenie projektów na repozytorium GIT znajdującym się pod adresem <https://bitbucket.org/ctpcluster>. Najlepszym pomysłem jest dodanie repozytorium z projektem i przesłanie plików projektowych z klastra na serwer bitbucket.

Tworzymy nowe repozytorium:



Rysunek 10: Tworzenie repozytorium

- w katalogu projektu na klastrze tworzymy nowy katalog o nazwie `git`, (`mkdir git`).
- kopujemy istotne pliki projektowe do katalogu `git` i inicjalizujemy repozytorium `git init`
- dodajemy adres repozytorium  
`git remote add origin ssh://git@bitbucket.org/ctpcluster/` (nazwa repozytorium)
- dodajemy pliki `git add *`
- wykonujemy commit `git commit -m "Pierwszy commit. Hello World z instrukcji"`
- przesyłamy pliki z klastra do repozytorium `git push -u origin master`

Sprawozdanie proszę wysłać pod adres [ctpcluster@gmail.com](mailto:ctpcluster@gmail.com).

## 4 Wstęp do programowania równoległego w OpenMPI

W tym rozdziale zajmiemy się programowaniem równoległym na klastrze wykorzystując bibliotekę OpenMPI. Skrót MPI oznacza Message Passing Interface czyli w wolnym tłumaczeniu "Środowisko do przesyłu wiadomości". Warto podkreślić że programowanie, które wykorzystuje OpenMPI jest programowaniem równoległym gdzie każdy uruchomiony proces ma własną niezależną pamięć (stertę), na danym komputerze/węźle. Dlatego nie należy mylić programowania na systemie rozproszonym jak np. klaster z programowaniem na procesor wielordzeniowy, gdzie procesory zwykle współdzielą część pamięci i trzeba to uwzględnić przy programowaniu. Zajmiemy się teraz trzema programami napisanymi w języku C. C++ i Fortran też jest obsługiwany.

- Hello World
- Obliczenie liczby  $\pi$
- Mnożenie macierzy

Zacznijmy od podstaw programowania w OpenMPI, kompilator nosi nazwę `mpicc` dla języka C. Aby program wiedział, że będzie uruchamiany równolegle na kilku komputerach musimy mu o tym powiedzieć. Wykonyje się to poprzez podanie pliku `hostfile` który specyfikuje węzły w klastrze obliczeniowym komendzie `mpirun` jako parametr. Plik `hostfile` w naszym przypadku wygląda tak:

```

localhost slots=1
slave1
slave2
slave3
slave4
slave5
slave6
slave7

```

Parametr `slots` określa ilość rdzeni procesora. W naszym przypadku jest to po prostu 1.

## 4.1 Hello World

Przykładowe HelloWorld napisane przy wykorzystaniu biblioteki OpenMPI:

```

/* Source http://www.slac.stanford.edu/comp/unix/farm/mpi.html */
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);                                /* starts MPI */
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);             /*get number of processes*/
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);                  /*get current process id */
    MPI_Get_processor_name(processor_name, &namelen);      /*get processor name*/

    printf("Process %d on %s out of %d\n", rank, processor_name, numprocs);

    MPI_Finalize();
}

```

Kompilacja programu:

`mpicc hello_world.c -o hello_world.exe`

Uruchomienie (na 8 węzłach):

`mpirun -np 8 --hostfile hostfile ./hello_world.exe`

W tym przypadku plik `hostfile` znajduje się w tym samym katalogu co uruchamiany program.

Wynik uruchomienia tego programu to:

```

[ Read 9 lines ]

cluster@master:~/software/hello_world_example$ mpirun -np 8 --hostfile ~/.mpi_hostfile hello_world.exe
Process 0 on master out of 8
Process 2 on slave2 out of 8
Process 1 on slave1 out of 8
Process 4 on slave4 out of 8
Process 5 on slave5 out of 8
Process 6 on slave6 out of 8
Process 3 on slave3 out of 8
Process 7 on slave7 out of 8
cluster@master:~/software/hello_world_example$ 

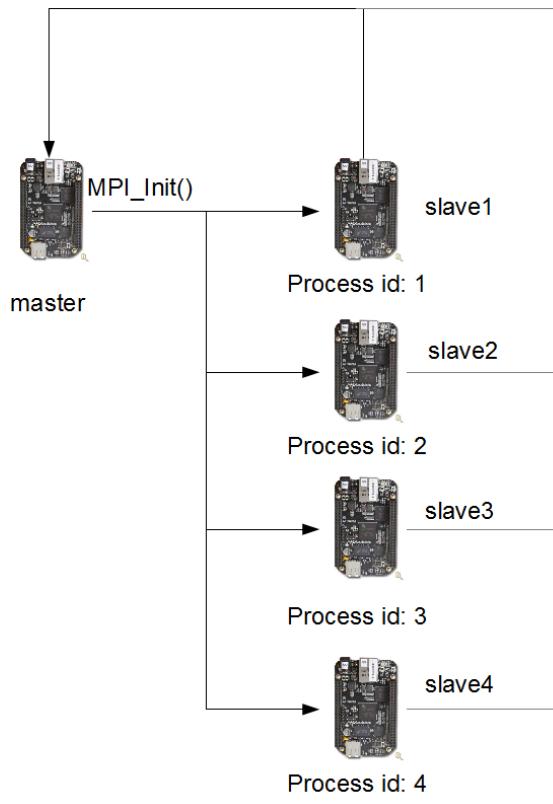
```

Rysunek 11: Wynik uruchomienia programu `hello_world.exe`

Jak widać każdy z węzłów wyświetlił swoją linijkę. Co ciekawe kolejność w jakiej otrzymamy poszczególne wiadomości jest losowa. Przejdźmy teraz do wytlumaczenia do czego służą poszczególne funkcje.

- `MPI_Init()` uruchamia MPI
- `MPI_Comm_size()` pobiera liczbę procesów i przypisuje tę wartość do zmiennej
- `MPI_Comm_rank()` pobiera ID procesu (numer) i przypisuje tę wartość do zmiennej
- `MPI_Get_processor_name()` pobiera nazwę komputera i zapisuje ją do tablicy typu char
- `MPI_Finalize()` kończy pracę MPI i terminuje program

Podkreślę jeszcze raz, aby zrozumieć na jakiej zasadzie działa ten program należy sobie wyobrazić iż wszystkie instrukcje pomiędzy funkcjami `MPI_Init()`, `MPI_Finalize()` są **równolegle** uruchomione na każdym z komputerów. Ilustruje to poniższy obraz.



Rysunek 12: Równolegle uruchomienie procesów na klastrze

## 4.2 Obliczenie $\pi$

Policzmy  $\pi$  metodą Funkcji Dzeta Riemanna. Funkcja wygląda następująco  $\zeta = \sum_{n=1}^{\infty} \frac{1}{n^s}$ . Dzięki wartości  $\zeta$  jesteśmy w stanie policzyć  $\pi$  wykonując operację  $\pi = \sqrt{6\zeta}$ . Obliczenie powyższej sumy na pojedynczym komputerze jest dosyć trywialne:

```

sum      = 0.0;
for(k=inf-1;k>=1;k--)
{
    sum += 1.0/pow(k, (double)n);
}
pi = sqrt(sum*6.0);
  
```

W jaki sposób wykorzystać nasz klaster do obliczenia  $\pi$  tą metodą? Skoro musimy policzyć sumę to najprostszym rozwiążaniem będzie rozdzielenie jej na  $n$  części, gdzie  $n$  to liczba komputerów w klastrze. Wystarczy policzyć nowe granice pętli i każdy z komputerów policzy jedną część sumy. Na końcu wszystkie prześlą swoje sumy cząstkowe do komputera *master* gdzie odbędzie się ostateczne sumowanie. Wygląda to następująco:

```

loop_min      = 1 +  (int)((tid + 0) * (inf-1)/NCPUs);
loop_max      =         (int)((tid + 1) * (inf-1)/NCPUs);

for(i=loop_max-1; i>=loop_min; i--)
{
    p_sum += 1.0/pow(i, (double)n);
}

MPI_Reduce(&p_sum, &sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

if (tid == 0)
{
    pi = sqrt(sum*6.0);
}

```

Gdzie `tid` jest numerem procesu, a `NCPUs` liczbą wszystkich komputerów w klastrze. Funkcja `MPI_Reduce()` przypisuje zmiennej `sum` sumy cząstkowe z każdego procesu `p_sum`.

Teoretycznie dzięki takiemu zabiegowi jesteśmy uzyskać N-krotne przyspieszenie obliczeń. W praktyce rzadko się to udaje. Pełne kody źródłowe powyższych programów znajdują się w katalogu `/home/cluster/software/`. Po komplikacji programy należy uruchomić w następujący sposób:

Wersja nie-równoległa:

`./rzf.exe -l 100000000 -n 2`

Czas obliczeń na jednym komputerze w klastrze 42s.

Wersja równoległa:

Kompilacja:

`mpicc rzf_2.c -o rzf_2.exe`

Uruchomienie:

`mpirun -np 8 --hostfile hostfile ./rzf_2.exe -l 100000000 -n 2`

Czas obliczeń na ośmiu komputerach 5.2s.

### 4.3 Mnożenie macierzy

Ostatnim przykładem jakim przyjrzymy się bliżej jest mnożenie macierzy. Najważniejsze aby zrozumieć możliwości komunikacji między węzłami w klastrze. Tak jak w poprzednim przykładzie poznaliśmy funkcję `MPI_Reduce()` tak teraz poznamy dwie funkcje `MPI_Send()` i `MPI_Recv()`.

**Algorytm mnożenia macierzy** Mnożenie macierzy jest popularnym zagadnieniem, istnieje wiele algorytmów mnożenia macierzy biorących pod uwagę ich specyficzne właściwości i inne cechy, my jednak zajmiemy się najprostszym bezpośrednim algorymem. Niestety złożoność obliczeniowa bezpośredniego algorytmu bardzo szybko rośnie wraz z rozmiarem macierzy -  $O(n^3)$ .

```

/*
 *   c[i][j] = a_row[i] dot b_col[j]   for all i,j
 *           a_row[i] -< a[i][0 .. DIM-1]
 *           b_col[j] -< b[0 .. DIM-1][j]
 *
 */
for(i=0; i<DIM; i++)
{
    for(j=0; j<DIM; j++)
    {
        dot=0.0;
        for(k=0; k<DIM; k++)
            dot += a[i][k]*b[k][j];
        c[i][j]=dot;
    }
}

```

```
}
```

Na pierwszy rzut oka widać, że wewnętrzną sumę w głównej pętli możemy zredukować na tej samej zasadzie jak w przykładzie z obliczeniem  $\pi$ .

```
loop_min    = (int) ((long) (tid + 0) *  
                  (long) (DIM) / (long) NCPUs);  
loop_max    = (int) ((long) (tid + 1) *  
                  (long) (DIM) / (long) NCPUs);  
  
for(i=loop_min;i<loop_max;i++)  
{  
    for(j=0;j<DIM;j++)  
    {  
        dot=0.0;  
        for(k=0;k<lt;DIM;k++)  
            dot += a[i][k]*b[k][j];  
        c[i][j]=dot;  
    }  
  
}
```

Aby przesyłać informacje między węzłami należy skorzystać z funkcji `MPI_Send()` i `MPI_Recv()`. Gdy węzeł typu *slave* skończy obliczenia to przesyła dane do węzła *master*. Bardzo istotne jest aby ilość wywołań funkcji `MPI_Send()` i `MPI_Recv()` zgadzała się, w przeciwnym wypadku program zablokuje się przy uruchomieniu, wystąpi tzw. *deadlock*.

```

for(i=1;i<NCPUs;i++)
{
    /* have the master process receive the row */
    loop_min = (int)((long)(i + 0) *
                      (long)(DIM)/(long)NCPUs);
    loop_max = (int)((long)(i + 1) *
                      (long)(DIM)/(long)NCPUs);

    /* receive the rows from the i_th_ process to the master process */
    if (tid == 0)
    {
        for(k=loop_min;k<loop_max;k++)
        {
            MPI_Recv(c[k],DIM,MPI_DOUBLE,i,i*k,
                     MPI_COMM_WORLD,&stat);
        }
    }

    /* send the rows from the i_th_ process to the master process */
    if (tid == i)
    {
        for(k=loop_min;k<loop_max;k++)
        {
            MPI_Send(c[k],DIM,MPI_DOUBLE,0,i*k,
                     MPI_COMM_WORLD);
        }
    }
}

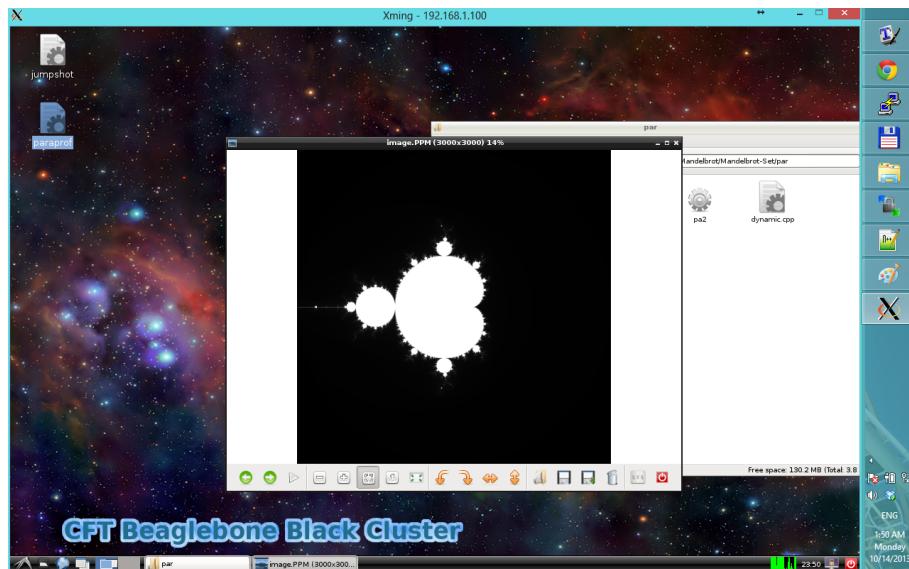
```

Nie jest to optymalne aby tak wysyłać dane z i do węzła. Jednak ten przykład ma zadanie pokazać możliwości komunikacji w bibliotece OpenMPI. Aby przyspieszyć pracę programu można dodatkowo zastosować tzw. *loop unrolling*.

**Uruchomienie i porównanie wydajności** Pełne kody źródłowe programów mnożenia macierzy znajdują się w katalogu /home/cluster/software/matrix\_mult. Czas potrzebny do przemnożenia macierzy  $N = 1024$  na jednym węźle wynosi: ok. 85s. Na osmiu: 11 s.

#### 4.4 Przykłady zaawansowanych programów OpenMPI

W katalogu /home/cluster/software/other znajdują się inne ciekawe programy, takie jak np. obliczenie i render Zbioru Mandelbrota.

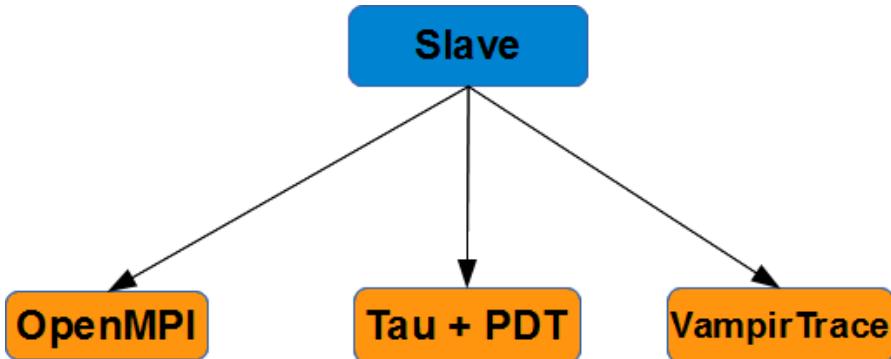


Rysunek 13: Profilowanie hello\_world przy pomocy paraprof

## 5 Dla zaawansowanych - profilowanie programów w OpenMPI

Na każdym węźle typu slave znajduje się oprogramowani służące do profilowania napisanych programów w OpenMPI:

- VampirTrace - oprogramowanie testujące wydajność programów uruchamianych na klastrze
- TAU i PDT - dodatkowe oprogramowanie profilujące z możliwością pracy w trybie graficznym



Rysunek 14: Oprogramowanie na węzłach klastra

Węzeł *master* poza oprogramowaniem, które występuje na węzłach typu *slave*, zawiera dodatkowo:

- powłokę graficzną LXDE
- serwer NFS

**Ważna uwaga** - do poprawnego profilowania programów wymagana jest synchronizacja czasu na każdym z węzłów.

### 5.1 Vampirtrace

Program VampirTrace pozwala na wylistowanie w jakiej funkcji program spędził najwięcej czasu, co daje możliwość szukania wąskich gardeł w wydajności programów równoległych. Istnieją cztery możliwości zastosowania VampirTrace do profilowania aplikacji: manual, source, binary, oraz runtime. Więcej informacji można znaleźć w User's Manual [?].

**Profilowanie hello\_world** Sprawdźmy, która funkcja wykonuje się najczęściej. Zastosujemy teraz profilowanie binarne.

```
|| mpirun -np 8 --hostfile ~/.mpi_hostfile vtrun -mpi ./hello_world.exe
```

Wynikiem powinno być:

Rysunek 15: Profilowanie hello\_world przy pomocy VampirTrace

Na załączonym zrzucie ekranu widać listę zmiennych środowiskowych, zmienne zaczynające się od VT\* służą do konfiguracji VampirTrace. Po tej operacji wygenerowany został szereg plików.

hello_world.0.def	hello_world.4.events	hello_world.exe
hello_world.1.events	hello_world.5.events	hello_world.otf
hello_world.1.stats	hello_world.6.events	hello_world.prof.txt
hello_world.2.events	hello_world.7.events	hello_world.thumb
hello_world.3.events	hello_world.8.events	hostfile

Mogliśmy wygenerować plik \*.tex, który będzie zawierał te same informacje które zostały wyświetlane w konsoli.

```
|| otfprofile-mpi -i hello_world.otf
```

Powyższa komenda generuje plik *result.tex* który można zamienić na plik PDF, komendą *pdflatex*. **Istotna uwaga** Zgodnie z dokumentacją powinna być możliwa dokładniejsza generacja informacji wraz z wykresami [?]. Analogicznie wykonuje się profilowanie źródłowe. Tylko tym razem należy skompilować projekt wykorzystując VampirTrace.

```
|| vtcc -vt:cc mpicc hello_world.c -o hello_world_vt.exe
```

I uruchomić, ale w **normalny sposób**:

```
|| mpirun -np 8 --hostfile ~/.mpi_hostfile ./hello_world_vt.exe
```

## 5.2 Tau

Program TAU pozwala na graficzną prezentację działania programu wykorzystującego OpenMPI. Program *hello\_world.c* komplujemy wykorzystując skrypt *tau\_cc.sh*

```
|| tau_cc.sh hello_world.c -o hello_world_tau.exe
```

Uruchamiamy normalnie:

```
|| mpirun -np 8 --hostfile ~/.mpi_hostfile ./hello_world_tau.exe
```

Przy uruchomieniu mogą wystąpić komunikaty o braku BFD. Następnie wykonujemy polecenia:

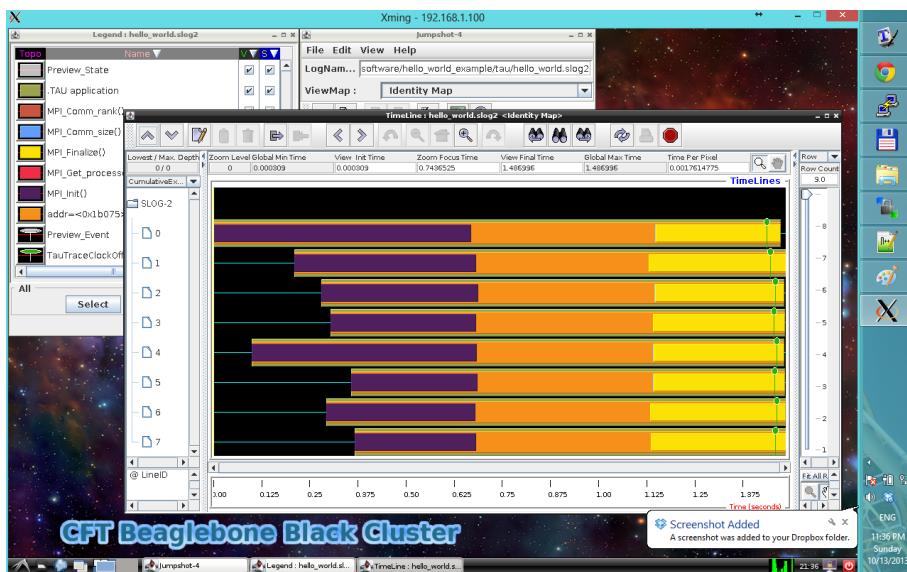
```
tau_treemerge.pl  
tau2slog2 tau.trc tau.edf -o hello_world.slog2  
tau2otf tau.trc tau.edf hello_world.otf  
trace2profile tau.trc tau.edf
```

Po wykonaniu powyższych kroków powinny zostać wygenerowane następujące pliki:

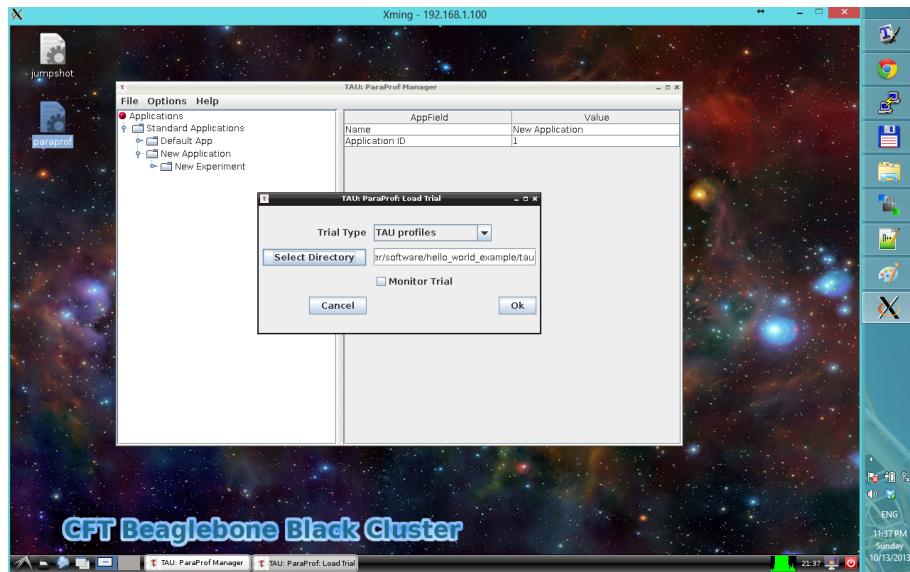
```
events.0.edf      hello_world.1.events    profile.2.0.0  
tautrace.2.0.0.trc          hello_world.c        profile.3.0.0  
tautrace.3.0.0.trc          hello_world.o        profile.4.0.0  
tautrace.4.0.0.trc          hello_world.otf       profile.5.0.0  
tautrace.5.0.0.trc          hello_world.slog2     profile.6.0.0  
tautrace.6.0.0.trc          hello_world_tau.exe profile.7.0.0  
tautrace.7.0.0.trc  
events.6.edf      hostfile                  tau.edf           tau.trc  
events.7.edf      profile.0.0.0            tautrace.0.0.0.trc  
hello_world.0.def    profile.1.0.0            tautrace.1.0.0.trc
```

Korzystając z programu *otfprofile* można analogicznie jak dla VampirTrace wygenerować plik \*.tex. Zajmijmy się jednak graficzną reprezentacją wykorzystując programy *paraprof* oraz *jumpshot*.

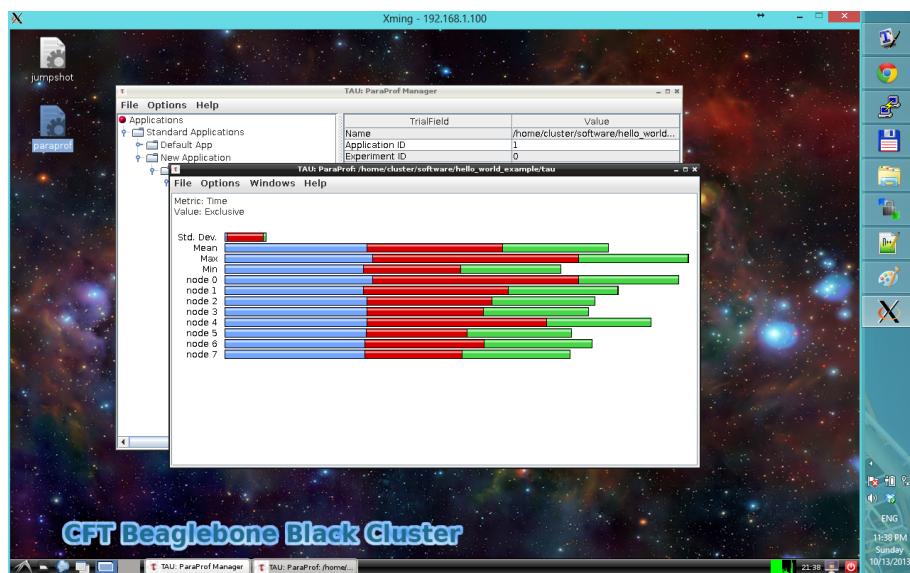
**Paraprof i Jumpshot** Uruchamiamy graficzny interfejs klastra, np. za pomocą programu Xming. Na pulpicie znajdują się programy *paraprof* oraz *jumpshot*. Uruchamiamy je i nawigujemy do odpowiedniego katalogu gdzie skompilowany został program przy pomocy TAU. Program *jumpshot* potrzebuje pliku \*.slog2 do wygenerowania wykresów. Aby wykorzystać *paraprof* należy otworzyć katalog z plikami \*.profile z opcją TAU PROFILE.



Rysunek 16: Profilowanie hello\_world przy pomocy Jumpshot



Rysunek 17: Profilowanie hello\_world przy pomocy paraprof



Rysunek 18: Profilowanie hello\_world przy pomocy paraprof

## 6 Podsumowanie

Klaster zbudowany w Centrum Fizyki Teoretycznej PAN jest urządzeniem dostępnym dla każdego zainteresowanego programowaniem równoległym. Pozwala on na uruchamianie programów napisanych z wykorzystaniem biblioteki OpenMPI: w języku C/C++ oraz Fortran oraz profilowanie tych programów przy pomocy narzędzi TAU i Vampirtrace. Dzięki niemu możliwa jest nauka pracy z programami napisanymi w OpenMPI oraz ze strukturą prostego superkomputera. Na koniec pragnę zachęcić do wykonywania projektów i do zadawania pytań w razie jakichkolwiek problemów czy niejasności z powyższą instrukcją/poradnikiem.

## 7 Zmiany wersji

- poprawienie błędów
- dodanie opisu dostępu do repozytorium GIT

## Literatura

- [1] BeagleBone Official Site <http://beagleboard.org/Products/BeagleBone%20Black>
- [2] Spider Robot zbudowany z wykorzystaniem Beaglebone <http://www.youtube.com/watch?v=JXyewd98e9Q>
- [3] Strona domowa programu PuTTY <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
- [4] Podstawy Linuxa film <http://www.youtube.com/watch?v=cfLx1FxNGuc>
- [5] 10 rzeczy które należy wpoić użytkownikowi Linuxa <http://www.ubucentrum.net/2011/12/podstawy-10-rzeczy-ktore-musisz-wpoic.html>
- [6] Wiki Fedory - Podstawy Linuxa [http://wiki.fedora.pl/wiki/Podstawy\\_Linuksa](http://wiki.fedora.pl/wiki/Podstawy_Linuksa)
- [7] [http://www.tu-dresden.de/die\\_tu\\_dresden/zentrale\\_einrichtungen/zih/forschung/projekte/vampirtrace/dateien/VT-UserManual-5.14.4.pdf](http://www.tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/vampirtrace/dateien/VT-UserManual-5.14.4.pdf)
- [8] Open-MPI <http://www.open-mpi.org/>
- [9] MPI in Thirty Minutes <http://www.linux-mag.com/id/5759/>
- [10] Xming Download <http://sourceforge.net/projects/xming/>
- [11] Wikipedia: Beowulf Cluster [http://en.wikipedia.org/wiki/Beowulf\\_cluster](http://en.wikipedia.org/wiki/Beowulf_cluster)
- [12] Wikipedia: Unix Shell [http://en.wikipedia.org/wiki/Unix\\_shell](http://en.wikipedia.org/wiki/Unix_shell)
- [13] Raspberry Pi <http://www.raspberrypi.org/>
- [14] Rasberry Pi Cluster example 1 <http://www.zdnet.com/build-your-own-supercomputer-out-of-rasberry-pi-cluster-example-1>
- [15] Raspberry Pi Cluster example 2 <http://www.sciencedaily.com/releases/2013/11/131113092128.htm>
- [16] Przykład profilowania aplikacji za pomocą TAU <http://goo.gl/RqYoxl>