



**СУПЕРКОМПЬЮТЕРНЫЙ**  
КОНСОРЦИУМ УНИВЕРСИТЕТОВ РОССИИ



**МОЛОДЕЖНАЯ ШКОЛА**  
**«Суперкомпьютерные технологии**  
**в науке, промышленности и образовании»**

**«Есть ли жизнь после МРІ?»**

***Абрамов С.М.***

***д.ф.-м.н., чл.-корр. РАН,***

***ИПС имени А.К.Айламазяна***

***РАН***

**ННГУ им. Н.И. Лобачевского, Нижний Новгород, 26-31 октября**

**2009 г.**





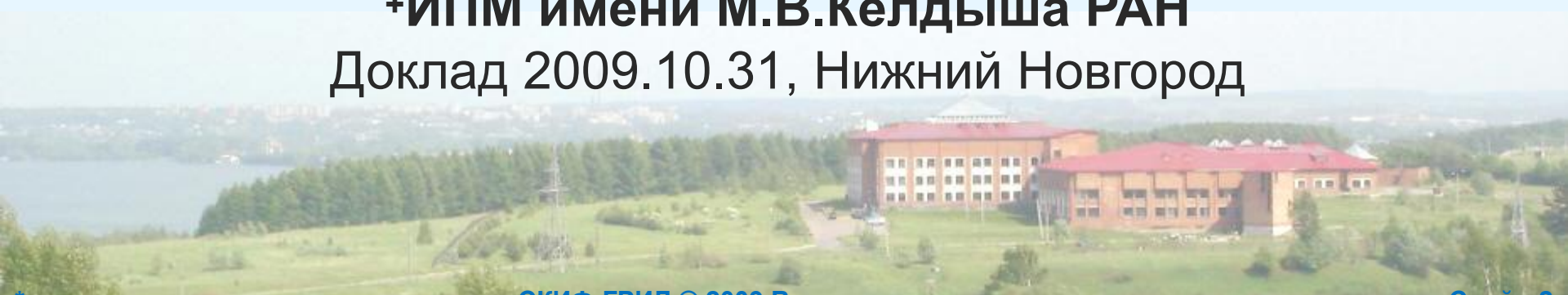
# Есть ли жизнь после МРІ?

Абрамов С.М.<sup>†</sup>, Климов А.В.<sup>‡</sup>,  
Лацис А.О.<sup>‡</sup>, Московский А.А.<sup>†</sup>

<sup>†</sup>ИПС имени А.К.Айламазяна РАН

<sup>‡</sup>ИПМ имени М.В.Келдыша РАН

Доклад 2009.10.31, Нижний Новгород





Переславль-Залесский.  
Институт программных систем  
имени А.К.Айламазяна  
Российской академии наук





# Переславль-Залесский



- ★ Красивый старинный (860 лет) город России на берегу Плещеева озера
- ★ Центр Золотого кольца
- ★ Родина Св.Александра Невского, родина многих великих князей
- ★ Здесь Петр Великий создавал свою первую «потешную флотилию» — место рождения Российского флота
- ★ Древний центр Российской Православной Церкви



# ИПС имени А.К.Айламазяна РАН, Переславль-Залесский





## Основание Института

- ★ Основан в 1984 году по постановлению ВПК для развития информатики и вычислительной техники в стране
- ★ Первый директор (1984–2003) — проф. А.К.Айламазян
- ★ В декабре 2008 Институту присвоено имя А.К.Айламазяна





## 2009: Организационная структура института



★ Исследовательский центр искусственного интеллекта



★ Исследовательский центр медицинской информатики

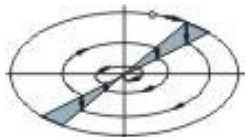


★ Исследовательский центр мультипроцессорных систем



★ Исследовательский центр системного анализа

★ Исследовательский центр процессов управления



★ Научно-образовательный центр —  
Международный детский компьютерный лагерь  
(МДКЦ) имени А.К.Айламазяна





# Университет города Переславля имени А.К.Айламазяна





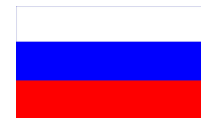


# Программы «СКИФ» и «СКИФ-ГРИД»



## Заказчики-координаторы

- НАН Беларуси
- Агентство «Роснаука»



## Головные исполнители

- Объединенный институт проблем информатики НАН Беларуси
- Институт программных систем РАН



## Исполнители

- «СКИФ» 2000-2004 — 10+10 организаций Беларуси и России
- «СКИФ-ГРИД» 2007-2010 — 10+20 организаций Беларуси и России



2003-2008: 5 суперЭВМ семейства «СКИФ» в рейтинге Top500

# Семейство суперЭВМ «СКИФ»: Ряды 1, 2, 3 и 4

Сделано: Ряды 1–3

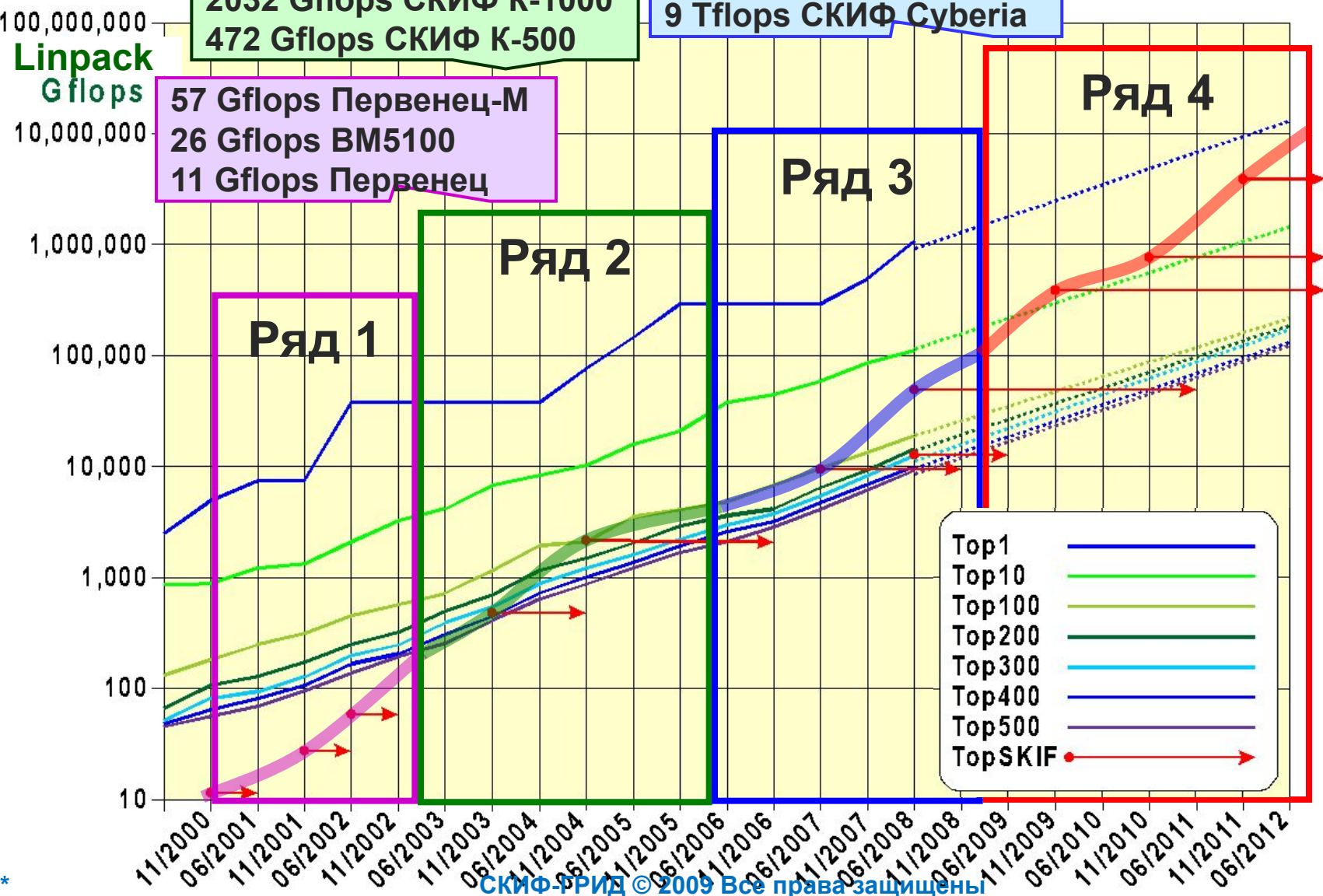
Ближайшие планы: Ряд 4

1 кв. 2012 СКИФ П~5.0  
 3 кв. 2010 СКИФ П-1.0  
 3 кв. 2009 СКИФ П-0.5

2032 Gflops СКИФ К-1000  
 472 Gflops СКИФ К-500

47.17 Tflops СКИФ МГУ  
 12.2 Tflops СКИФ Урал  
 9 Tflops СКИФ Cyberia

57 Gflops Первенец-М  
 26 Gflops ВМ5100  
 11 Gflops Первенец



За все время только шесть созданных в России суперЭВМ вошли в Top500. Пять из шести— СКИФы!

10,000,000  
Gflops  
1,000,000

100,000

10,000

1,000

100

11/2001

06/2002

11/2002

06/2003

11/2003

06/2004

11/2004

06/2005

11/2005

06/2006

11/2006

06/2007

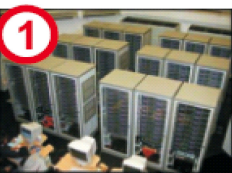
11/2007

06/2008

11/2008

06/2009

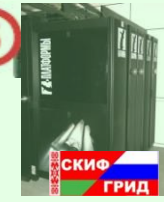
11/2009



1 2002 июнь  
MVS 1000M  
0.734/1.024 TFlops



3 2004 ноябрь  
СКИФ K-1000  
2.032/2.534 TFlops



5 2008 май  
СКИФ Урал  
12.2/15.9 TFlops



2 2003 ноябрь  
СКИФ K-500  
0.423/0.717 TFlops



4 2007 февраль  
СКИФ Cyberia  
9.013/12.002 TFlops



6 2008 май  
СКИФ МГУ  
47.1/60 TFlops



# Что затрудняет эффективное использование MPI в суперЭВМ ближайшего будущего?





## Проблемы MPI

- ★ Рост числа процессоров (и ядер) в суперЭВМ будет продолжаться
  - Сегодня 1 Pflops  $\approx$  20,000 CPU  $\approx$  80,000 ядер
  - Установки с 1,000,000 ядрами появятся очень скоро
- ★ Трудности эффективной реализации MPI для гигантского числа вычислительных узлов
  - да еще и многоядерных!
- ★ Трудности эффективного использования программистами MPI для случая гигантского числа вычислительных узлов
- ★ Есть разрыв между тем, что реализует аппаратура (SMP + односторонние обмены) и тем, что имеется в MPI



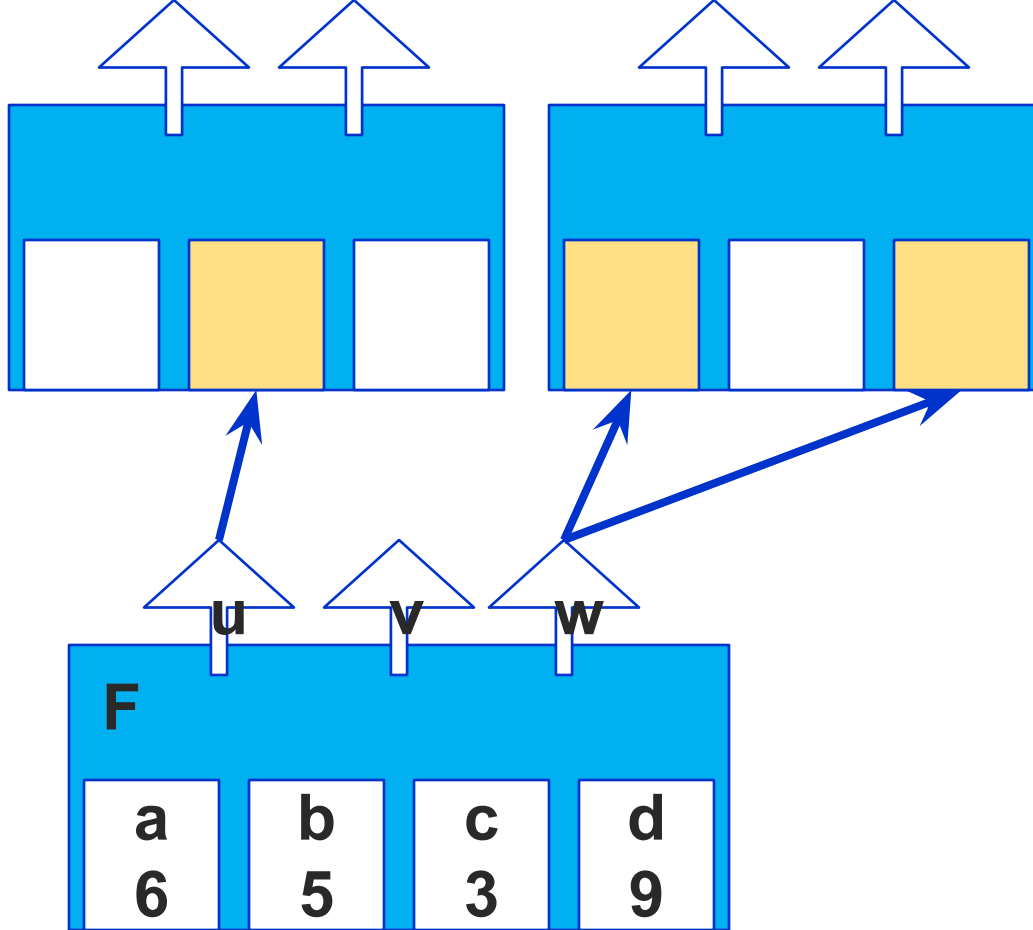
# **T-система: автоматическое динамическое распараллеливание программ**





## T-система (неформально)

- ★ Функциональная модель + императивное описание тела функции
- ★ Арность и коарность функций
- ★ Готовые и неготовые значения
- ★ Вызов T-функции — порождение процесса
- ★ Можно копировать неготовые значения, в том числе и передавать их как результат
- ★ Любые иные операции с неготовым значением приводит к «засыпанию» процесса на данной T-переменной
- ★ Побудка будет, когда T-переменная примет готовое значение
- ★ Состояние вычисления: сеть из процессов (ребра — отношение «поставщик-потребитель»), процесс



...

$$d = G(a, b)$$

$$b = d$$

$$u = a$$

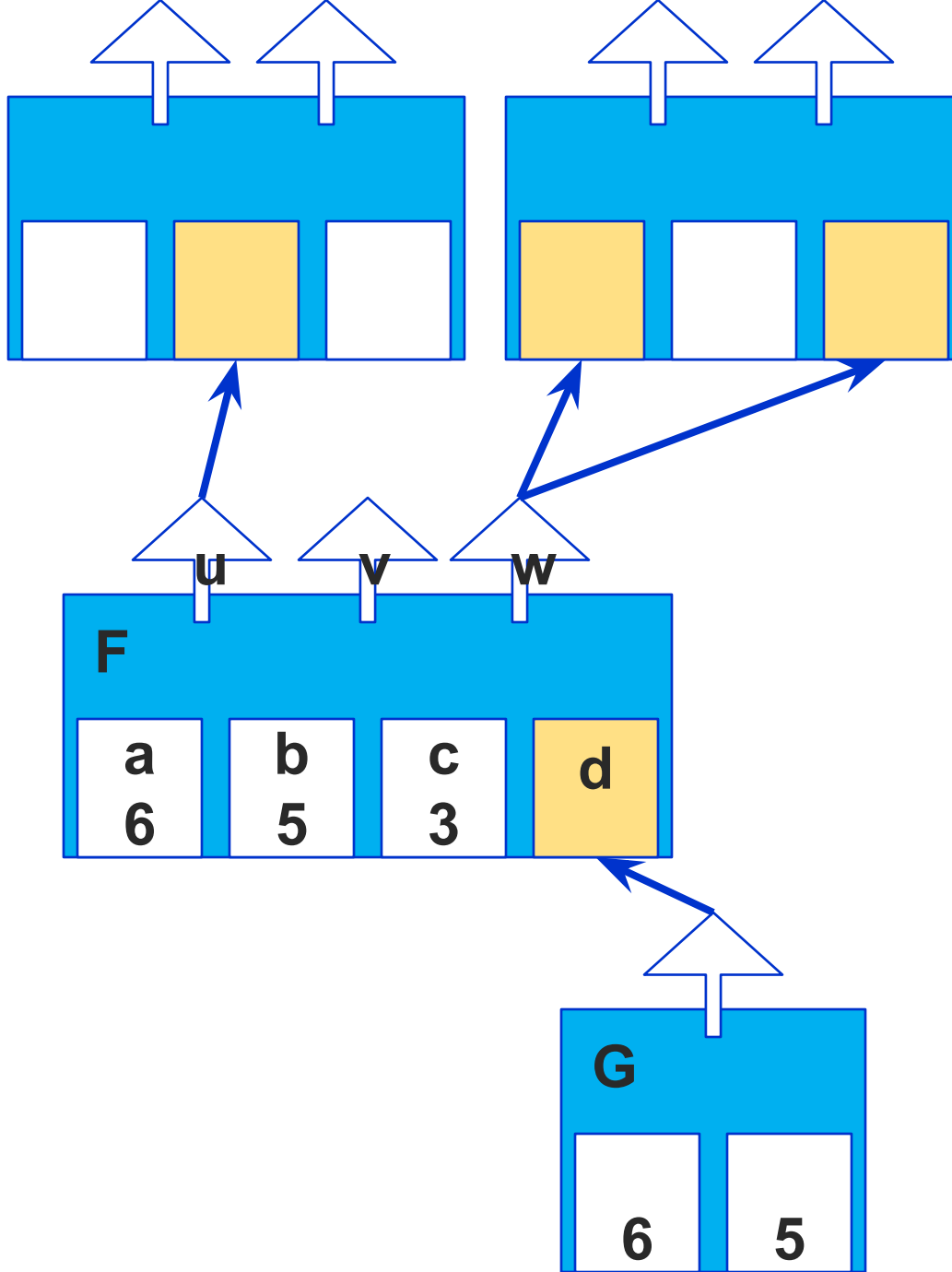
$$d = c$$

$$v = a$$

$$w = b$$

...





...

**$d = G(a, b)$**

$b = d$

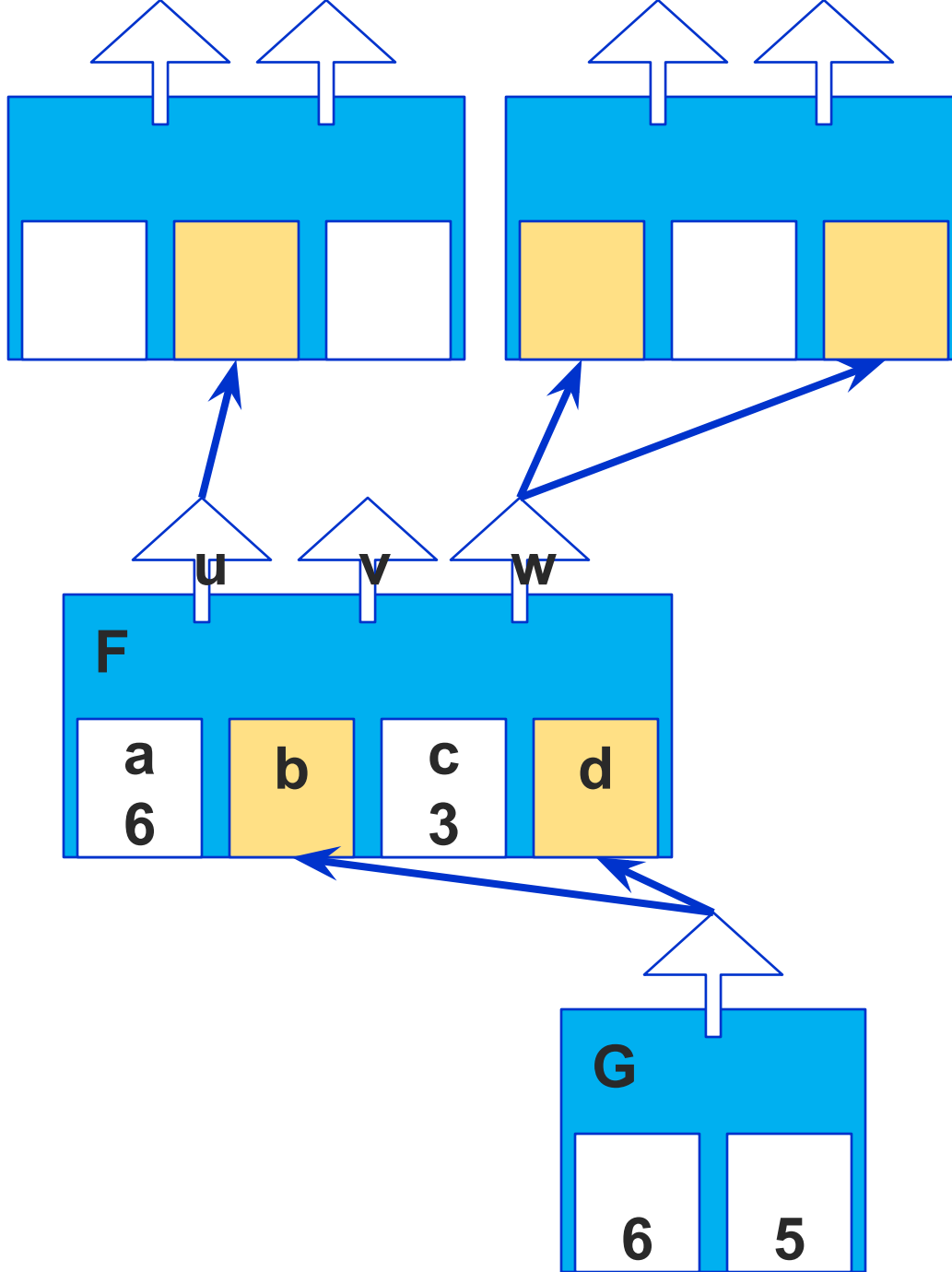
$u = a$

$d = c$

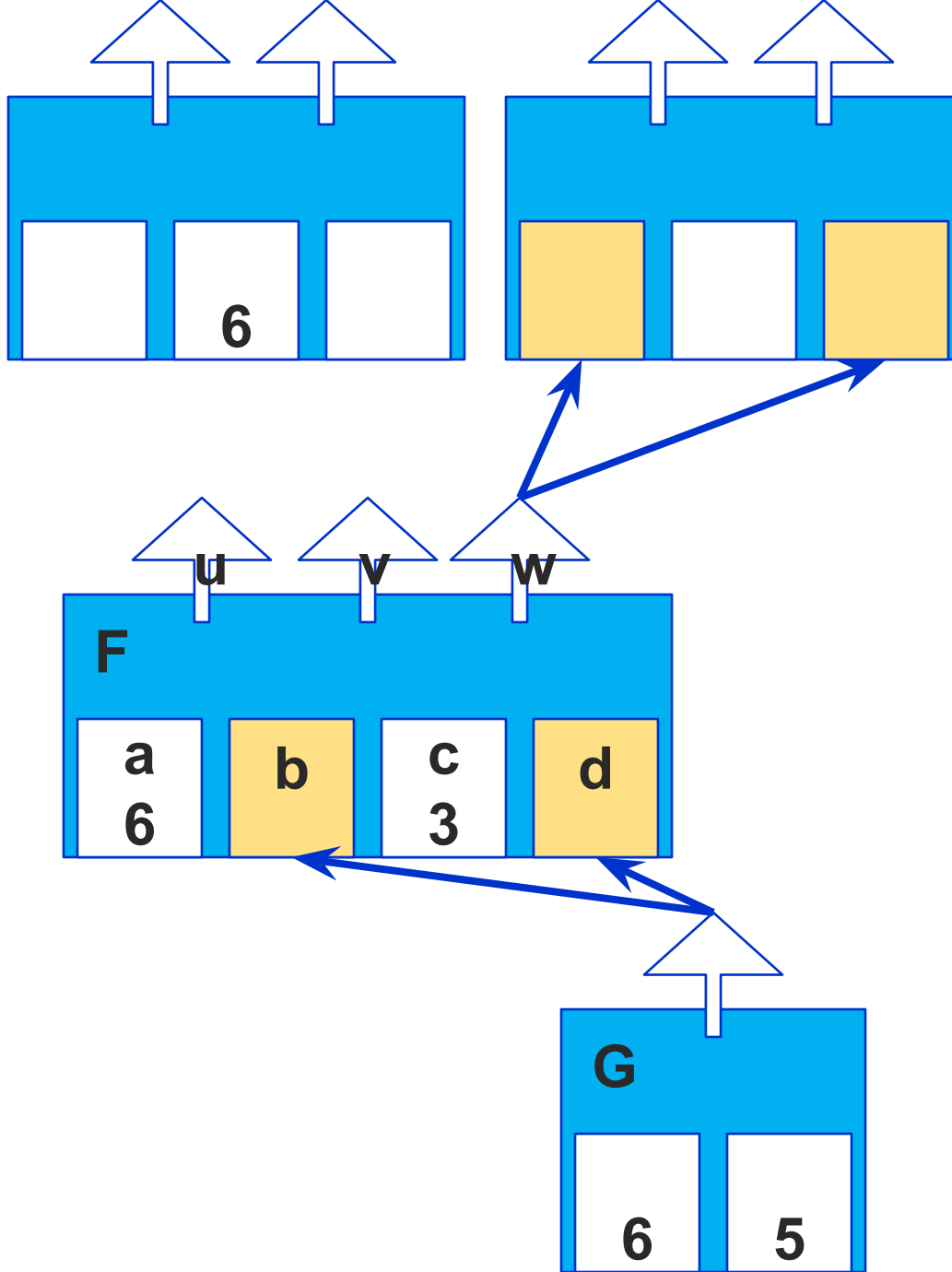
$v = a$

$w = b$

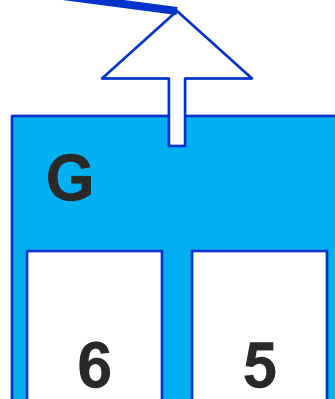
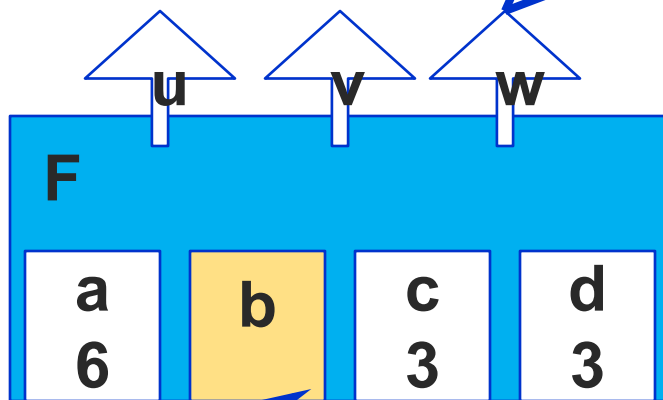
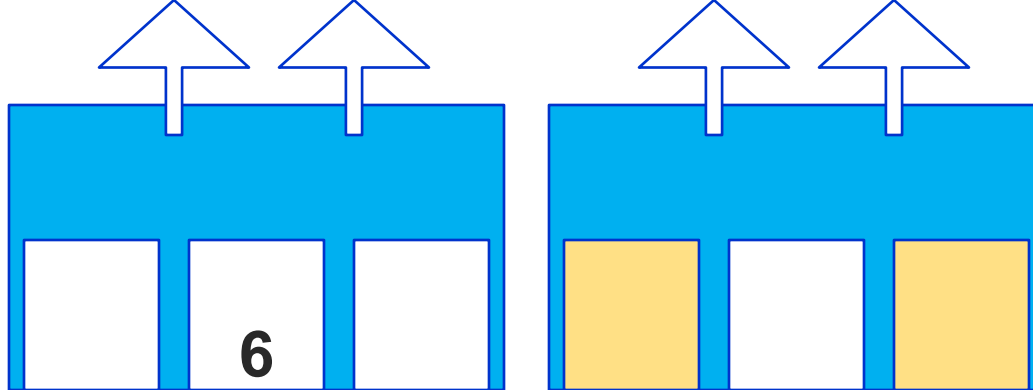
...



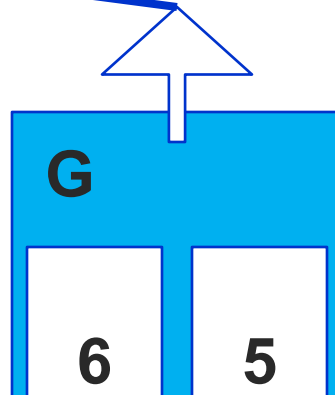
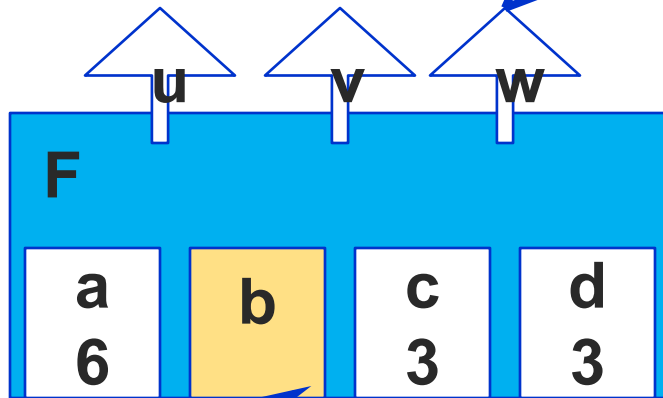
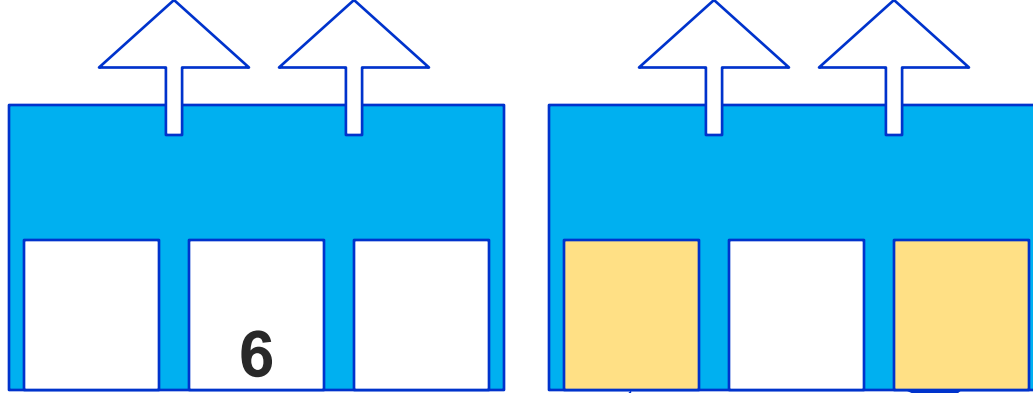
...  
 $d = G(a, b)$   
 **$b = d$**   
 $u = a$   
 $d = c$   
 $v = a$   
 $w = b$   
 ...



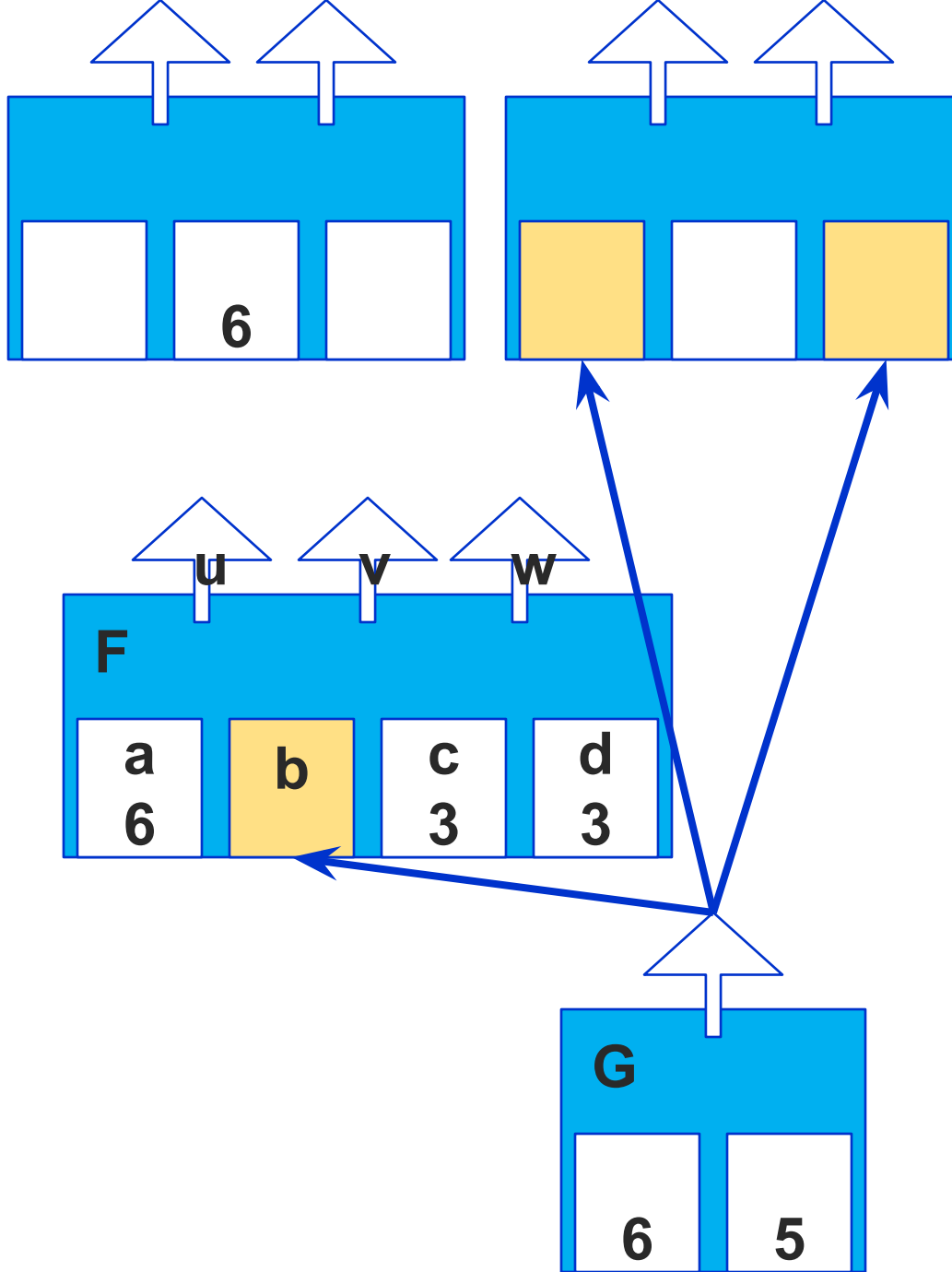
...  
 $d = G(a, b)$   
 $b = d$   
 **$u = a$**   
 $d = c$   
 $v = a$   
 $w = b$   
 ...



...  
 $d = G(a, b)$   
 $b = d$   
 $u = a$   
 $d = c$   
 $v = a$   
 $w = b$   
 ...

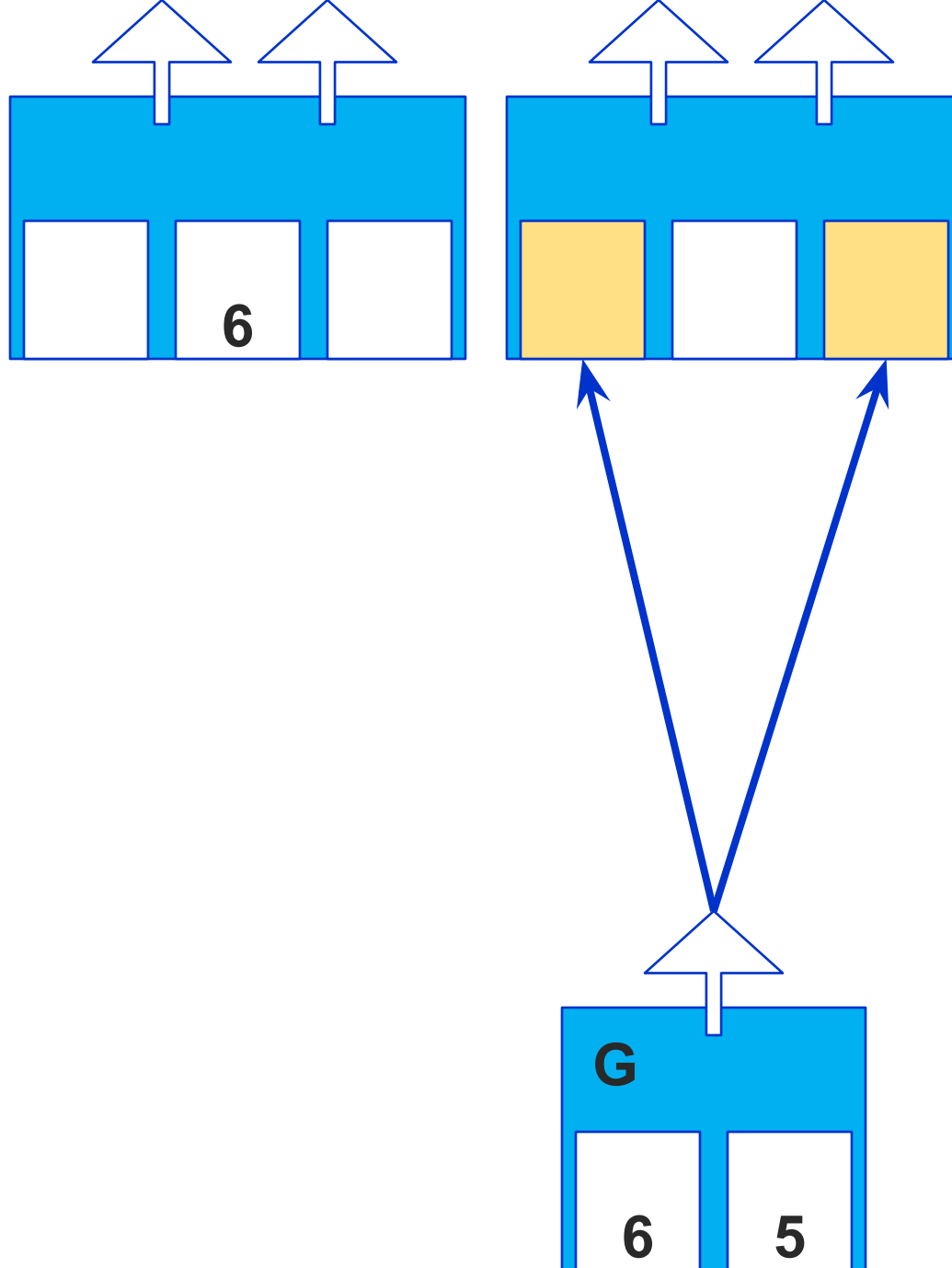


...  
 $d = G(a, b)$   
 $b = d$   
 $u = a$   
 $d = c$   
 $v = a$   
 $w = b$   
 ...



...  
 $d = G(a, b)$   
 $b = d$   
 $u = a$   
 $d = c$   
 $v = a$   
 $w = b$   
 ...

$w = b$





# T-System History

- ❑ **Mid-80-ies**  
Basic ideas of T-System
- ❑ **1990-ies**  
First implementation of T-System
- ❑ **2001-2002, "SKIF"**  
GRACE — Graph Reduction Applied to Cluster Environment
- ❑ **2003-current, "SKIF"**  
**Cooperation with Microsoft**  
Open TS — Open T-system





**Program Systems Institute  
Russian Academy of Sciences**

# **Open TS Overview**





# Comparison: T-System and MPI

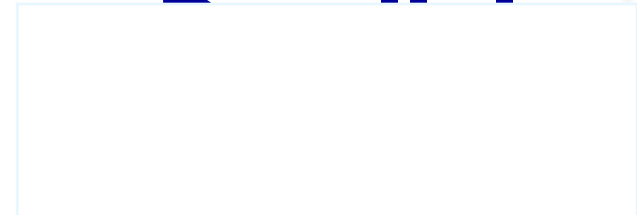
**High-level**  
a few  
keywords

**Low-level**  
hundred(s)  
primitives

**C/Fortran**

**Assembler**

**Sequential**





# T-System in Comparison

| <b>Related work</b>      | <b>Open TS differentiator</b>                    |
|--------------------------|--|
| Charm++                  | FP-based approach                                |
| UPC, mpC++               | Implicit parallelism                             |
| Glasgow Parallel Haskell | Allows C/C++ based low-level optimization        |
| OMPC++                   | Provides both language and C++ templates library |
| Cilk                     | Supports SMP, MPI, PVM, and GRID platforms       |



# Open TS: an Outline

- ❑ High-performance computing
- ❑ “Automatic dynamic parallelization”
- ❑ Combining functional and imperative approaches, high-level parallel programming
- ❑ T++ language: “Parallel dialect” of C++ — an approach popular in 90-ies



# T-Approach

- ❑ “Pure” functions (**tfunctions**) invocations produce grains of parallelism
- ❑ T-Program is
  - ★ Functional – on higher level
  - ★ Imperative – on low level (optimization)
- ❑ C-compatible execution model
- ❑ Non-ready variables, Multiple assignment
- ❑ “Seamless” C-extension (or Fortran-extension)



## T++ Keywords

- ❑ **tfun** — T-function
- ❑ **tval** — T-variable
- ❑ **tptr** — T-pointer
- ❑ **tout** — Output parameter (like &)
- ❑ **tdrop** — Make ready
- ❑ **twait** — Wait for readiness
- ❑ **tct** — T-context



**Program Systems Institute  
Russian Academy of Sciences**

# **Short Introduction (Sample Programs)**





# Sample Program (C++)

```
#include <stdio.h>
```

```
int fib (int n) {  
    return n < 2 ? n : fib(n-1)+ fib(n-2);  
}
```

```
int main (int argc, char **argv) {  
    if (argc != 2) { printf("Usage: fib <n>\n"); return 1; }  
    int n = atoi(argv[1]);  
    printf("fib(%d) = %d\n", n, fib(n));  
    return 0;  
}
```





# Sample Program (T++)

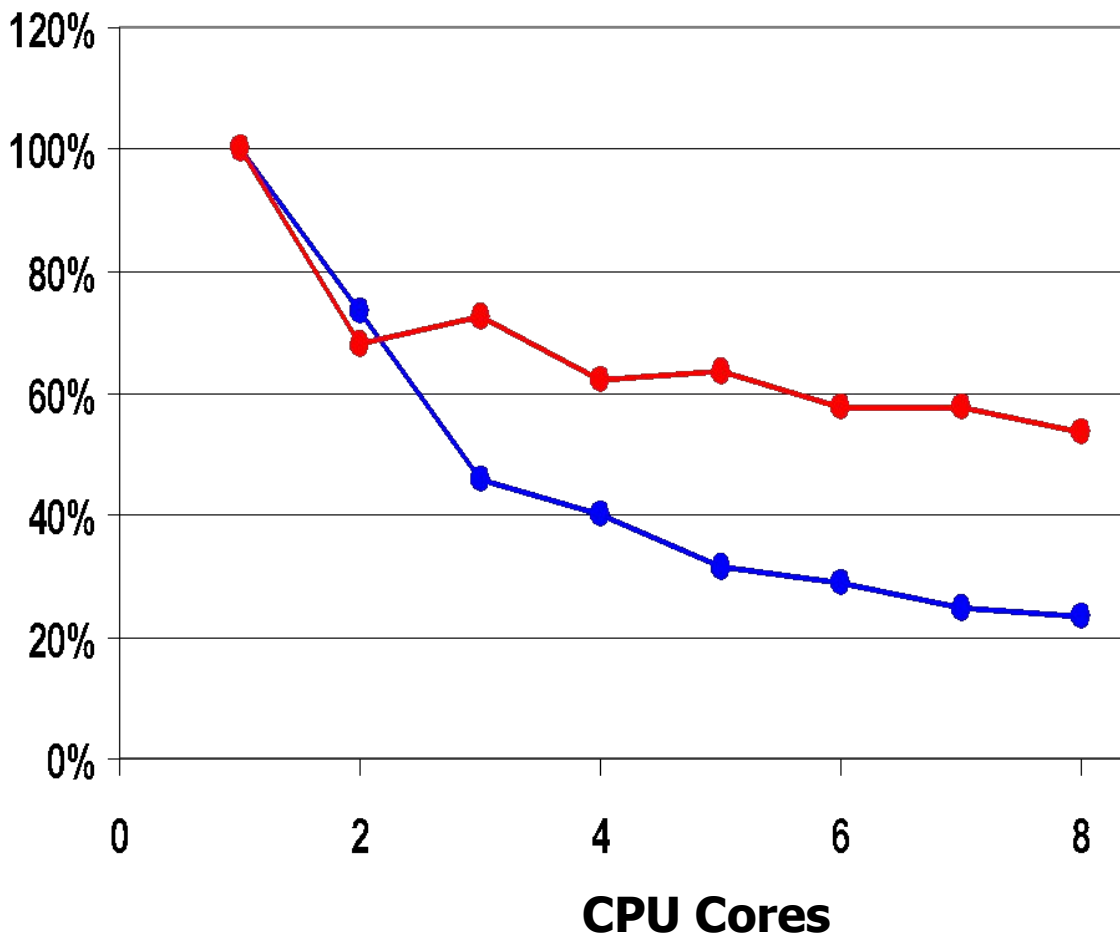
```
#include <stdio.h>
```

```
tfun int fib (int n) {  
    return n < 2 ? n : fib(n-1)+ fib(n-2);  
}
```

```
tfun int main (int argc, char **argv) {  
    if (argc != 2) { printf("Usage: fib <n>\n"); return 1; }  
    int n = atoi(argv[1]);  
    printf("fib(%d) = %d\n", n, (int)fib(n));  
    return 0;  
}
```



# Sample Program (T++)



- WinCCS cluster, 4 nodes
- **CPU:** AMD Athlon 64 X2 Dual Core Processor 4400+ 2.21 GHz
- Gigabit Ethernet
- **time%** =  $\text{time}_{\text{tapp}}(N) / \text{time}_{\text{tapp}}(1)$
- **CoE** =  $1 / (n \times \text{time}\%)$

—●— Time(%) —●— CoE



# Approximate calculation of Pi (C++)

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
double
    isum(double begin, double
        finish, double d) {
    double dl = finish - begin;
    double mid =
        (begin + finish) / 2;

    if (fabs(dl) > d)
        return isum(begin, mid, d)
            + isum(mid, finish, d);
    return f(mid) * dl;
}
```

```
double f(double x) {
    return 4/(1+x*x);
}
int main(int argc, char* argv[]){
    unsigned long h;
    double a, b, d, sum;

    if (argc < 2) {return 0;}
    a = 0; b = 1; h = atol(argv[1]);
    d = fabs(b - a) / h;
    sum = isum(a, b, d);
    printf("PI is approximately
        %015.15lf\n", sum);
    return 0;
}
```



# Approximate calculation of Pi (T++)

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
tfun double
    isum(double begin, double
        finish, double d) {
    double dl = finish - begin;
    double mid =
        (begin + finish) / 2;

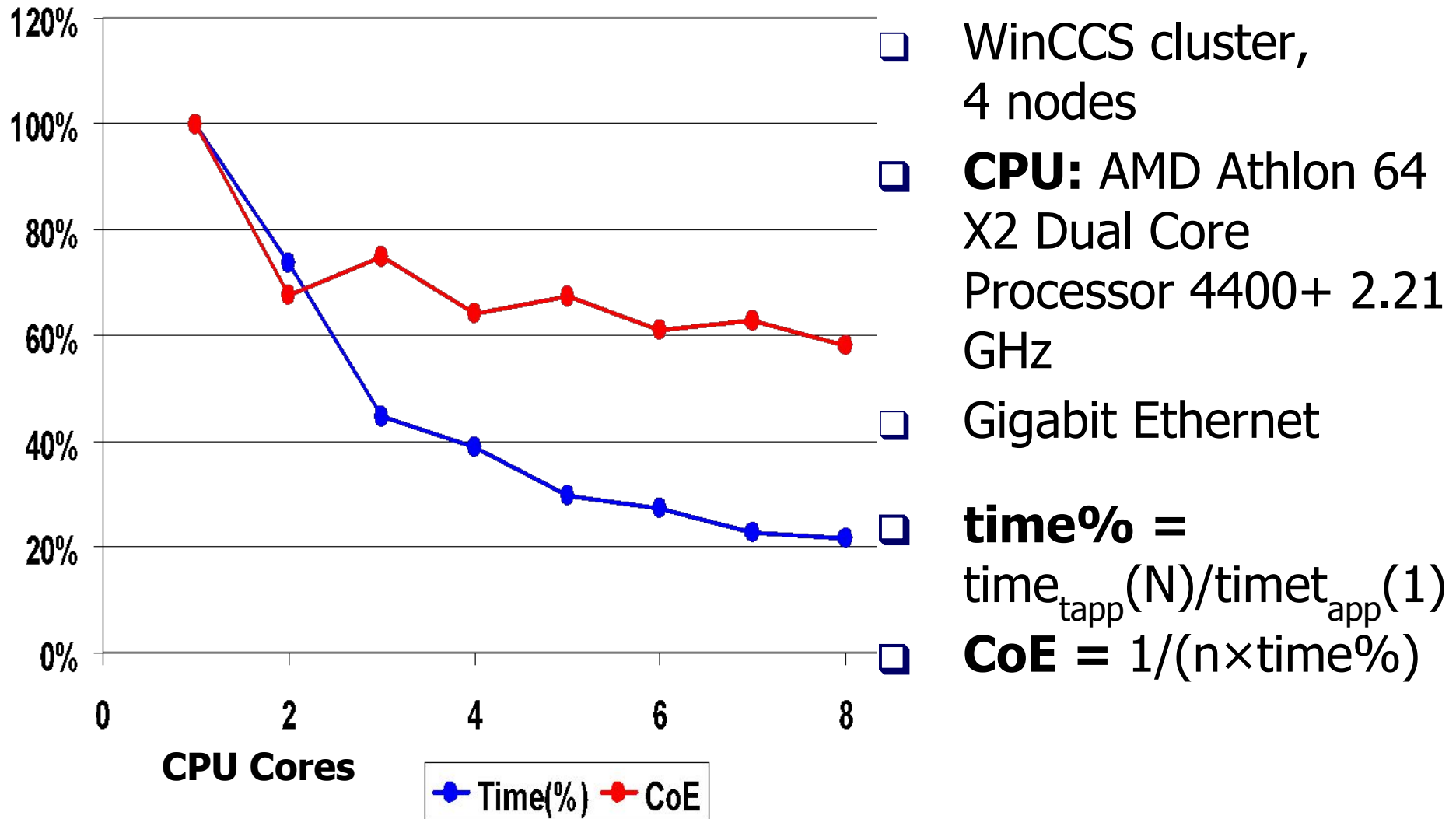
    if (fabs(dl) > d)
        return isum(begin, mid, d)
            + isum(mid, finish, d);
    return (double)f(mid) * dl;
}
```

```
tfun double f(double x) {
    return 4/(1+x*x);
}
tfun int main(int argc, char* argv[]){
    unsigned long h;
    double a, b, d, sum;

    if (argc < 2) {return 0;}
    a = 0; b = 1; h = atol(argv[1]);
    d = fabs(b - a) / h;
    sum = isum(a, b, d);
    printf("PI is approximately
        %015.15lf\n", sum);
    return 0;
}
```



# Calculation of Pi (T++)





# Map-Reduce

----- Original Message -----

**From:** [Alexy Maykov](#)

**Sent:** Monday, October 02, 2006 11:58 PM

**Subject:** MCCS projects

...

I work in Microsoft Live Labs ... I have several questions below:

1. How would you implement Map-Reduce in OpenTS?

...



## Map-Reduce (C++)

```
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>
#include <ctime>
using namespace std;

int fib (int n)
{
    return (n < 2) ? n : fib(n-1) + fib(n-2);
}

int plus (int val1, int val2)
{
    return val1 + val2;
}

int main (int argc, char *argv[ ])
{
    const int factor = 23;
    const int vector_size = 40;
    vector<int> a, b, c;
    vector<int> fa, fb;
```

```
    cout << " Filling vectors..." << endl;
    for (int i = 1; i <= vector_size; i++)
    {
        a.push_back(i % factor);
        b.push_back((vector_size + 1 - i) % factor);
        c.push_back(0);
        fa.push_back(0);
        fb.push_back(0);
    }

    cout << " Mapping..." << endl;
    transform(a.begin(), a.end(), fa.begin(), fib);
    cout << " Mapping..." << endl;
    transform(b.begin(), b.end(), fb.begin(), fib);
    cout << " Reducing..." << endl;
    transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
              ::plus);

    cout << endl << " Result: (" ;
    ostream_iterator<int> output(cout, " ");
    copy(c.begin(), c.end(), output);
    cout << "\\b)" << endl;

    return 0;
}
```



## Map-Reduce (C++)

```
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>
#include <ctime>
using namespace std;
```

```
int fib (int n)
{
    return (n < 2) ? n : fib(n-1) + fib(n-2);
}
```

```
int plus (int val1, int val2)
{
    return val1 + val2;
}
```

```
int main (int argc, char *argv[ ])
{
    const int factor = 23;
    const int vector_size = 40;
    vector<int> a, b, c;
    vector<int> fa, fb;
```

```
    cout << " Filling vectors..." << endl;
    for (int i = 1; i <= vector_size; i++)
    {
        a.push_back(i % factor);
        b.push_back(i % factor);
    }
```

**Fibonacci**

```
    cout << " Mapping..." << endl;
    transform(a.begin(), a.end(), fa.begin(), fib);
    cout << " Mapping..." << endl;
    transform(b.begin(), b.end(), fb.begin(), fib);
    cout << " Reducing..." << endl;
    transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
              ::plus);
```

```
    cout << endl << " Result: (" ;
    ostream_iterator<int> output(cout, " ");
    copy(c.begin(), c.end(), output);
    cout << endl;
```

**Just "Plus"**





# Map-Reduce (C++)

```
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>
#include <ctime>
using namespace std;
```

```
int fib (int n)
```

**Filling vectors:**

**a = [ k%23 | k ∈ [1..40]**

**b = [ (41-k)%23 | k ∈ [1..40]**

```
cout << " Filling vectors..." << endl;
for (int i = 1; i <= vector_size; i++)
```

```
{
  a.push_back(i % factor);
  b.push_back((vector_size + 1 - i) % factor);
  c.push_back(0);
  fa.push_back(0);
  fb.push_back(0);
}
```

```
cout << " Mapping..." << endl;
transform(a.begin(), a.end(), fa.begin(), fib);
```

```
cout << " Reducing..." << endl;
transform(b.begin(), b.end(), fb.begin(), fib);
```

```
cout << " Resulting..." << endl;
transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
```

```
cout << " Result: (" ;
```

**Five vectors: a, b, fa, fb, c**

```
return 0;
```

```
}
```



## Transform vectors:

```
fa = map fib a
```

```
fb = map fib b
```

```
c = zipWith plus fa fb
```

```
cout << " Filling vectors..." << endl;
for (int i = 1; i <= vector_size; i++)
    a[i] = fib(i);
for (int i = 1; i <= vector_size; i++)
    b[i] = fib(i);
for (int i = 1; i <= vector_size; i++)
    c[i] = plus(a[i], b[i]);
cout << " Mapping..." << endl;
transform(a.begin(), a.end(), fa.begin(), fib);
cout << " Mapping..." << endl;
transform(b.begin(), b.end(), fb.begin(), fib);
cout << " Reducing..." << endl;
transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
    ::plus);
cout << endl << " Result: (" ;
ostream_iterator<int> output(cout, " ");
copy(c.begin(), c.end(), output);
cout << "\b" << endl;
return 0;
}
```

```
#include <ctime>
using namespace std;
```

```
int fib (int n)
{
    return (n < 2) ? n : fib(n-1) + fib(n-2);
}
```

```
int plus (int val1, int val2)
{
    return val1 + val2;
}
```

```
int main (int argc, char *argv[ ])
{
    const int factor = 23;
    const int vector_size = 40;
    vector<int> a, b, c;
    vector<int> fa, fb;
```

```
cout << " Mapping..." << endl;
transform(a.begin(), a.end(), fa.begin(), fib);
cout << " Mapping..." << endl;
transform(b.begin(), b.end(), fb.begin(), fib);
cout << " Reducing..." << endl;
transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
    ::plus);
```

```
cout << endl << " Result: (" ;
ostream_iterator<int> output(cout, " ");
copy(c.begin(), c.end(), output);
cout << "\b" << endl;
```

```
return 0;
```

```
}
```



## Map-Reduce (C++)

```
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>
#include <ctime>
using namespace std;

int fib (int n)
{
    return (n < 2) ? n : fib(n-1) + fib(n-2);
}

int plus (int val1, int val2)
{
    return val1 + val2;
}

int main (int argc, char *argv[ ])
{
    const int factor = 23;
    const int vector_size = 40;
    vector<int> a, b, c;
    vector<int> fa, fb;
```

```
    cout << " Filling vectors..." << endl;
    for (int i = 1; i <= vector_size; i++)
    {
        a.push_back(i % factor);
        b.push_back((vector_size + 1 - i) % factor);
        c.push_back(0);
        fa.push_back(0);
        fb.push_back(0);
    }

    cout << " Mapping..." << endl;
    transform(a.begin(), a.end(), fa.begin(), fib);
    cout << " Mapping..." << endl;
    transform(b.begin(), b.end(), fb.begin(), fib);
    cout << " Reducing..." << endl;
    transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
              ::plus);

    cout << endl << " Result: (" ;
    ostream_iterator<int> output(cout, " ");
    copy(c.begin(), c.end(), output);
    cout << "\\b)" << endl;

    return 0;
}
```



## Map-Reduce (T++)

```
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>
#include <ctime>
using namespace std;
```

```
tfun int fib (int n)
{
    return (n < 2) ? n : fib(n-1) + fib(n-2);
}
```

```
tfun int plus (int val1, int val2)
{
    return val1 + val2;
}
```

```
tfun int main (int argc, char *argv[ ])
{
    const int factor = 23;
    const int vector_size = 40;
    vector<int> a, b, c;
    vector<tval int> fa, fb;
```

```
    cout << " Filling vectors..." << endl;
    for (int i = 1; i <= vector_size; i++)
    {
        a.push_back(i % factor);
        b.push_back((vector_size + 1 - i) % factor);
        c.push_back(0);
        fa.push_back(0);
        fb.push_back(0);
    }
```

```
    cout << " Mapping..." << endl;
    transform(a.begin(), a.end(), fa.begin(), fib);
    cout << " Mapping..." << endl;
    transform(b.begin(), b.end(), fb.begin(), fib);
    cout << " Reducing..." << endl;
    transform(fa.begin(), fa.end(), fb.begin(), c.begin(),
              ::plus);
```

```
    cout << endl << " Result: (" ;
```

**Vector of T-values**

```
    return 0;
}
```



# Map-Reduce (T++): "Laziness"

Filling, mapping — all T-functions are invoked, no T-Functions calculated: **0 seconds**

```
C:\WINDOWS\system32\cmd.exe
Running under unicomputer MP
[1.7Gf,381BM,1.49GiB]
Starting tfun main, good luc

09:15:30 Filling vectors..
09:15:30 Mapping...
09:15:30 Mapping...
09:15:30 Reducing...

09:15:38 Result: (2585 1599 990 615 385 246 165 123 110 123 165 246 385 615 990
1599 2585 4182 35422 28657 28657 35422 4182 2585 1599 990 615 385 246 165 123 11
0 123 165 246 385 615 990 1599 2585)

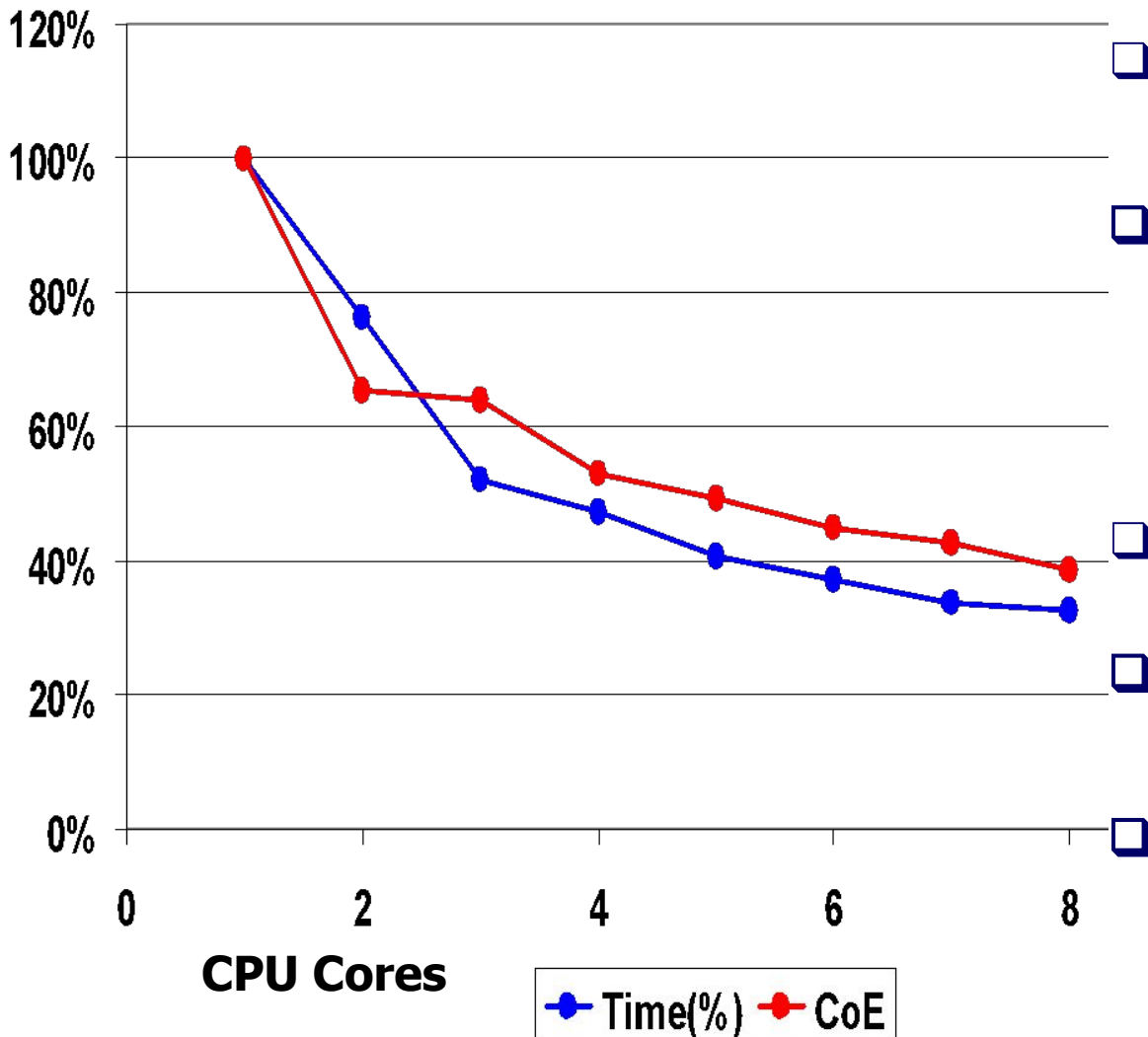
Tasks activated:      123253
Tasks exported:      0
Msgs sent:           0
Async Msgs:          0
Msgs size:           0
Taskboard visits:   123253
Scheduler time:      0.979
MPI time:            0.000
Idle time:           0.000
Tasks time:          3.580
Total time:          11.161

Press any key to continue . . .
```

Calculating of all T-functions, printing out: **8 seconds**



# Map-Reduce (T++)



- WinCCS cluster, 4 nodes
- **CPU:** AMD Athlon 64 X2 Dual Core Processor 4400+ 2.21 GHz
- Gigabit Ethernet
- **time%** =  $\text{time}_{\text{tapp}}(N) / \text{time}_{\text{tapp}}(1)$
- **CoE** =  $1 / (n \times \text{time}\%)$



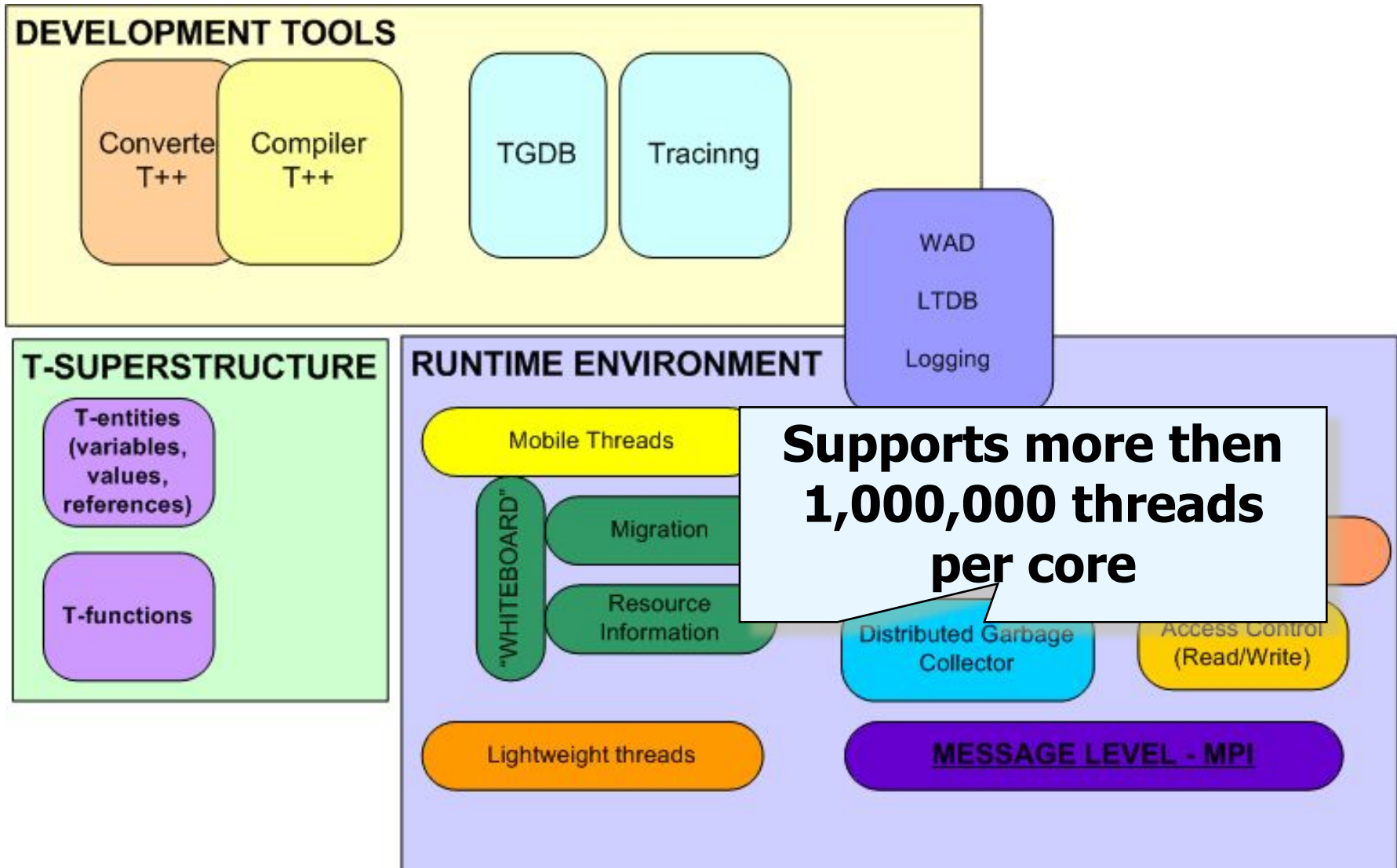
**Program Systems Institute  
Russian Academy of Sciences**

# **Inside OpenTS**





# Open TS: Environment







# Supermemory

- ❑ Utilization: non-ready values, resource and status information, etc.
- ❑ Object-Oriented Distributed shared memory (OO DSM)
- ❑ Global address space
- ❑ DSM-cell versioning
- ❑ On top - automatic garbage collection



# Multithreading & Communications

## ❑ Lightweight threads

- ★ PIXELS (1 000 000 threads)

## ❑ Asynchronous communications

- ★ A thread "A" asks non-ready value (or new job)
- ★ Asynchronous request sent: Active messages & Signals delivery over network to stimulate data transfer to the thread "A"
- ★ Context switches (including a quant for communications)

## ❑ Latency Hiding for node-node exchange



**Program Systems Institute  
Russian Academy of Sciences**

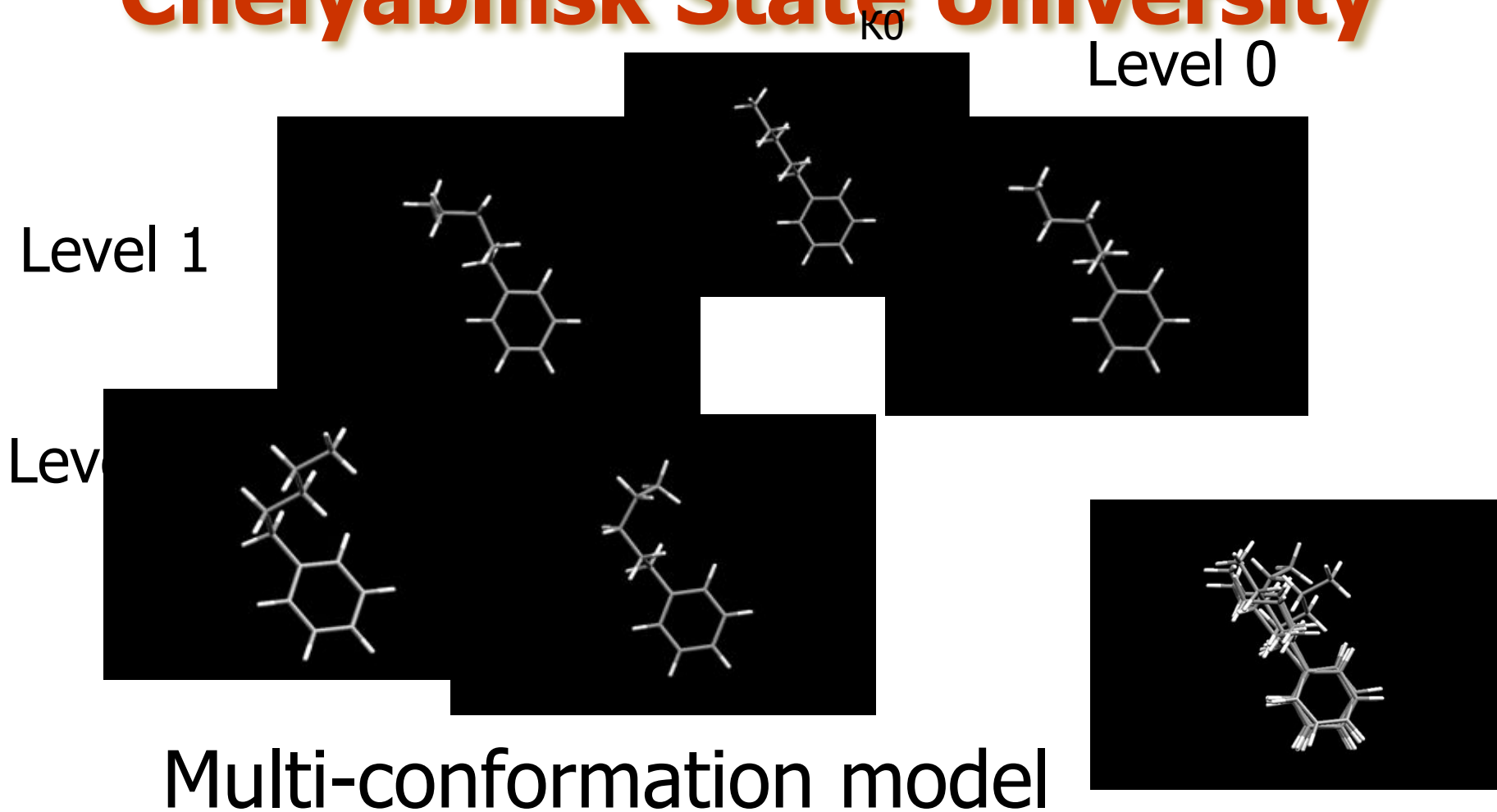
# **Open TS applications (selected)**





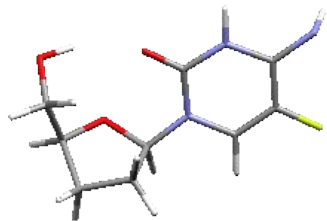
# MultiGen

## Chelyabinsk State University

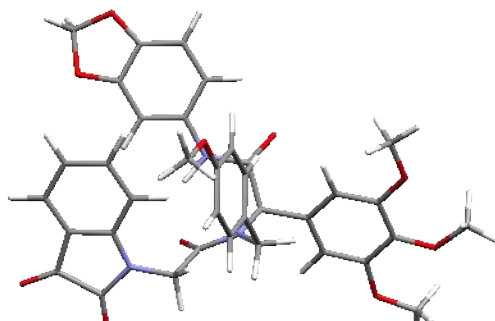




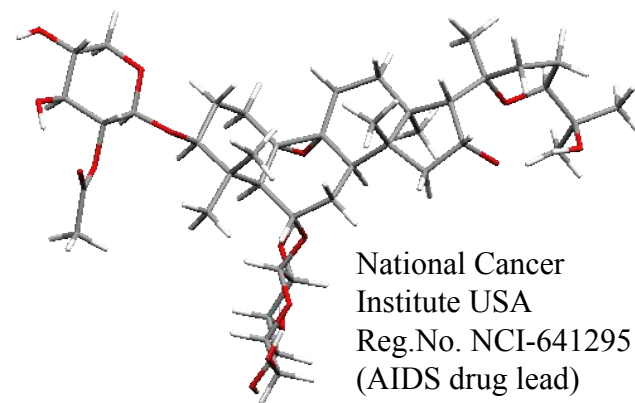
# MultiGen: Speedup



National Cancer Institute USA  
Reg.No. NCI-609067  
(AIDS drug lead)



TOSLAB company (Russia-Belgium)  
Reg.No. TOSLAB A2-0261  
(antiphlogistic drug lead)



National Cancer Institute USA  
Reg.No. NCI-641295  
(AIDS drug lead)

| Substance      | Atom number | Rotations number | Conformers | Execution time (min.:c) |         |          |
|----------------|-------------|------------------|------------|-------------------------|---------|----------|
|                |             |                  |            | 1 node                  | 4 nodes | 16 nodes |
| NCI-609067     | 28          | 4                | 13         | 9:33                    | 3:21    | 1:22     |
| TOSLAB A2-0261 | 82          | 18               | 49         | 115:27                  | 39:23   | 16:09    |
| NCI-641295     | 126         | 25               | 74         | 266:19                  | 95:57   | 34:48    |



# Aeromechanics

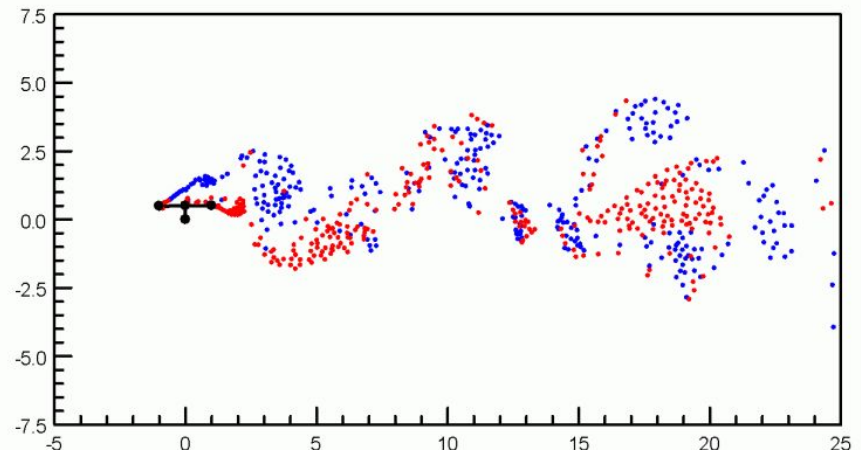
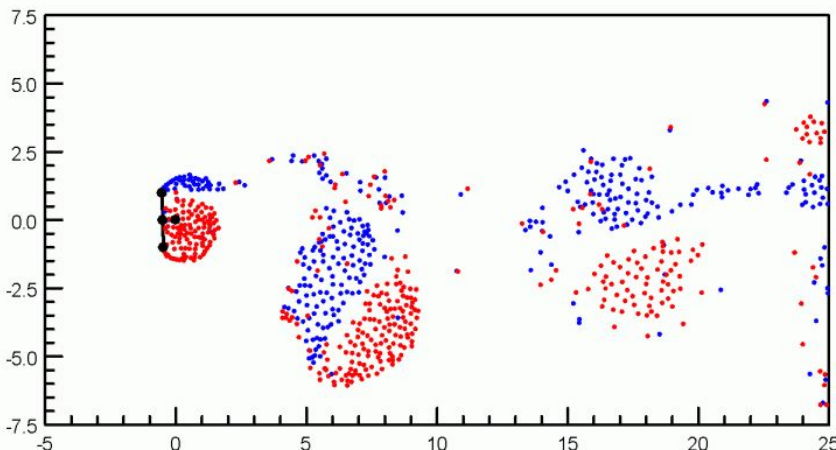
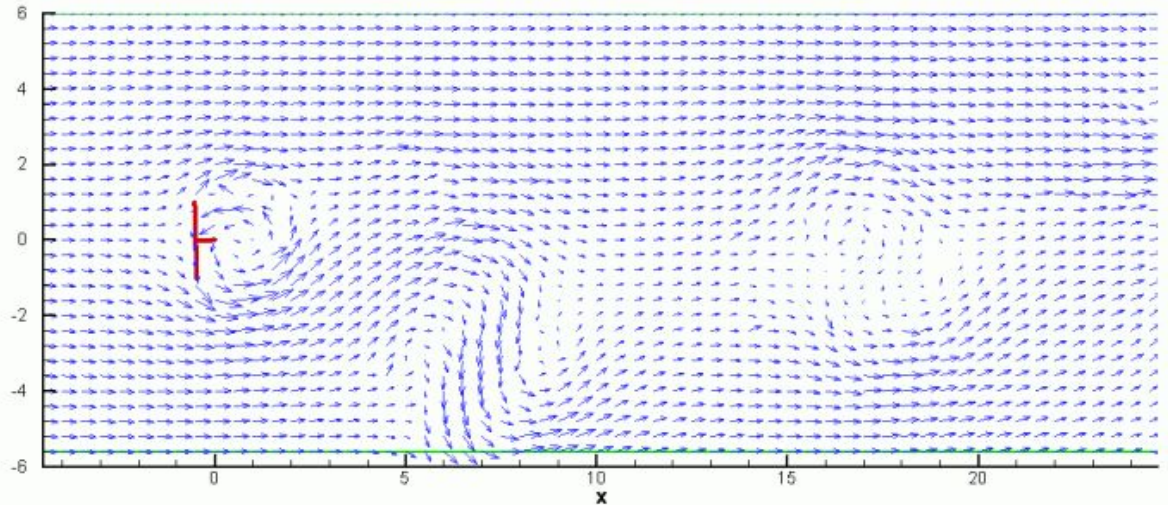
## Institute of Mechanics, MSU





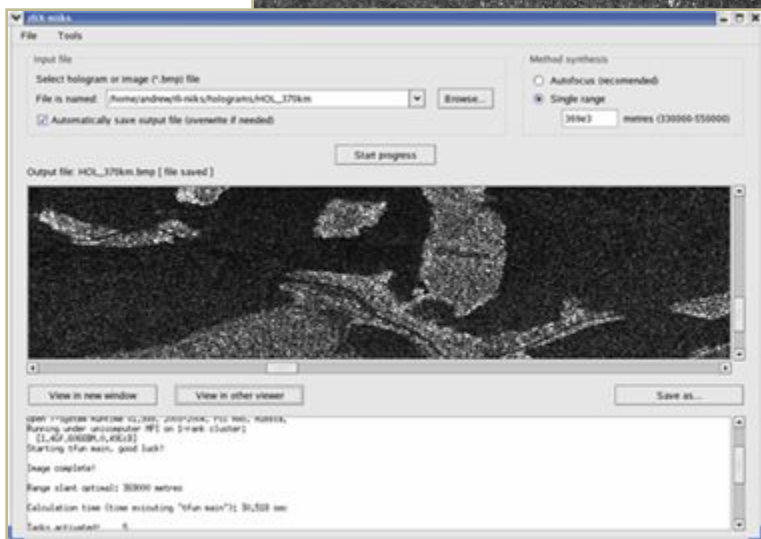
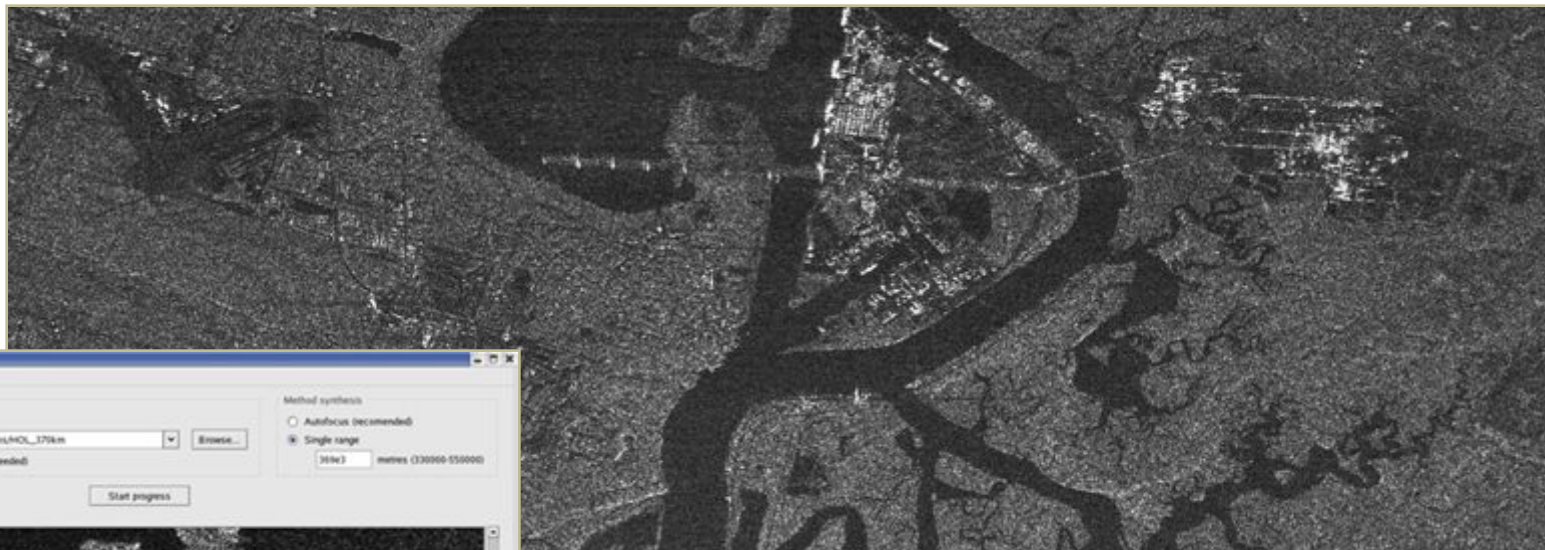
# Belocerkovski's approach

flow presented as a collection of small elementary whirlwind (colours: **clockwise** and **contra-clockwise** rotation)





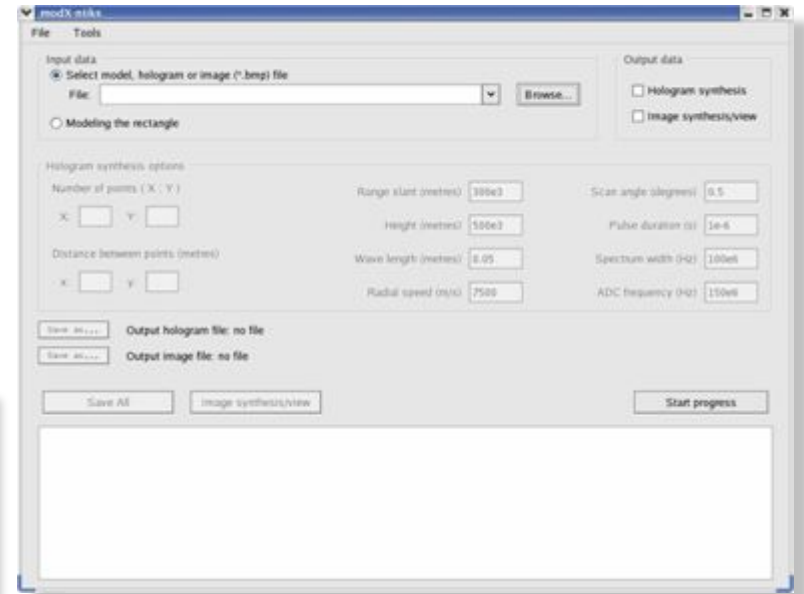
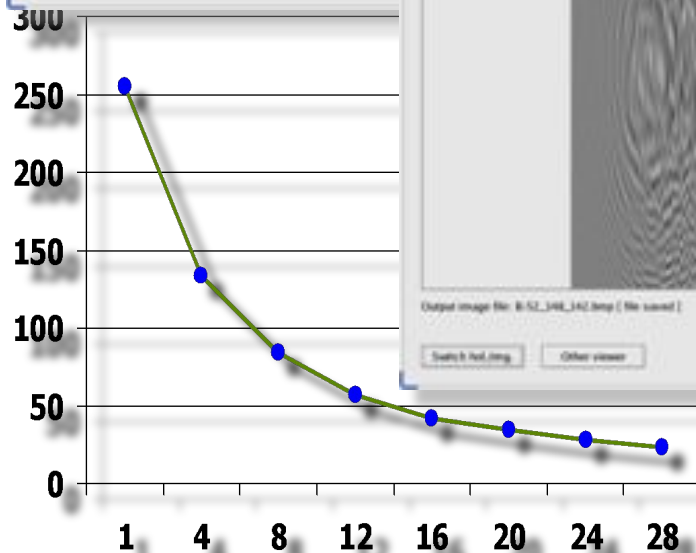
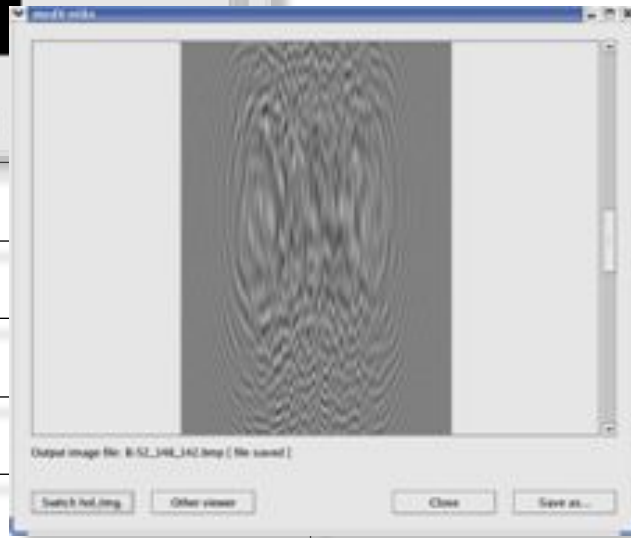
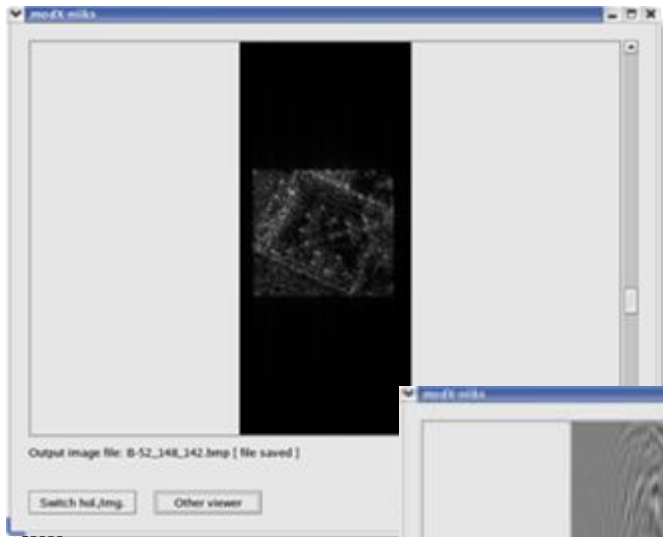
# Creating space-born radar image from hologram







# Simulating broadband radar signal

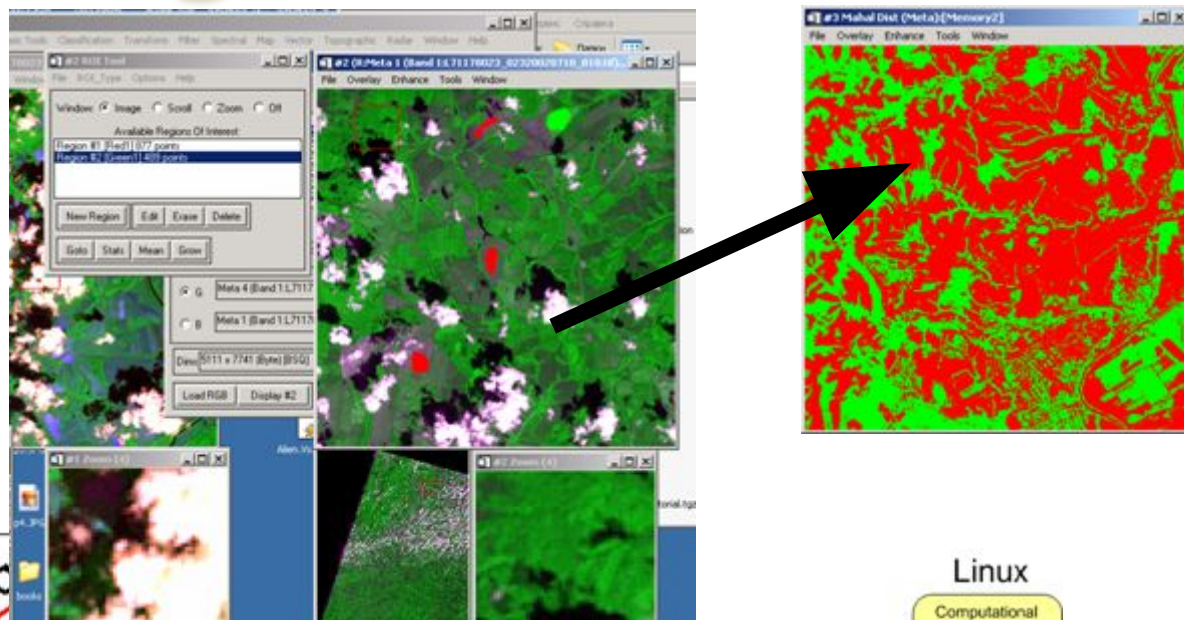


- ❑ Graphical User Interface
- ❑ Non-PSI RAS development team (Space research institute of Khrunichev corp.)

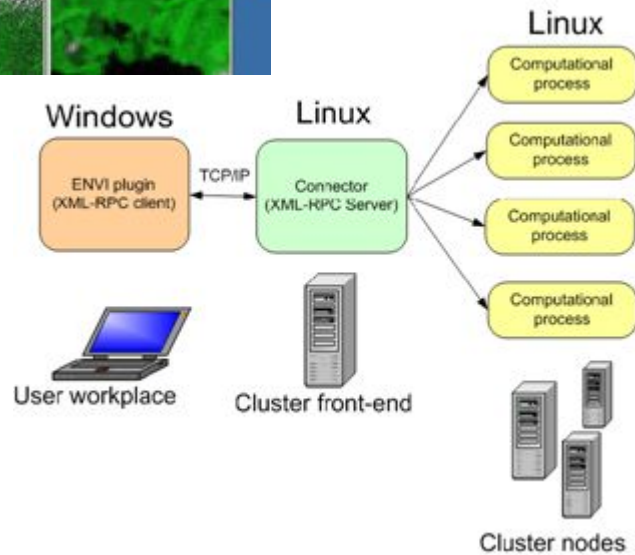
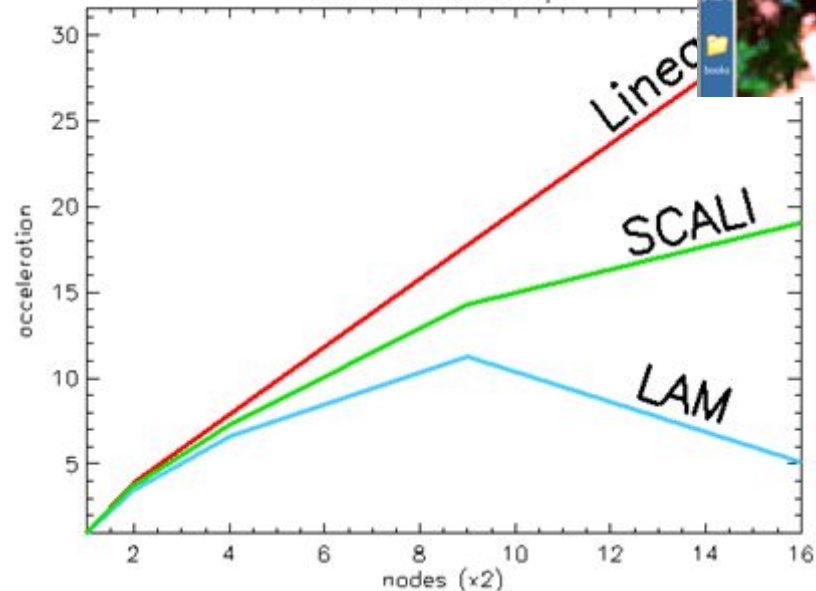


# Landsat Image Classification

- Computational "web-service"



Performance Graph





**Program Systems Institute  
Russian Academy of Sciences**

# **Open TS vs. MPI case study**





# Applications

- ❑ Popular and widely used
  - ❑ Developed by independent teams (MPI experts)
- 
- ❑ **PovRay** – Persistence of Vision Ray-tracer, enabled for parallel run by a patch
  - ❑ **ALCMD/MP\_lite** – molecular dynamics package (Ames Lab)



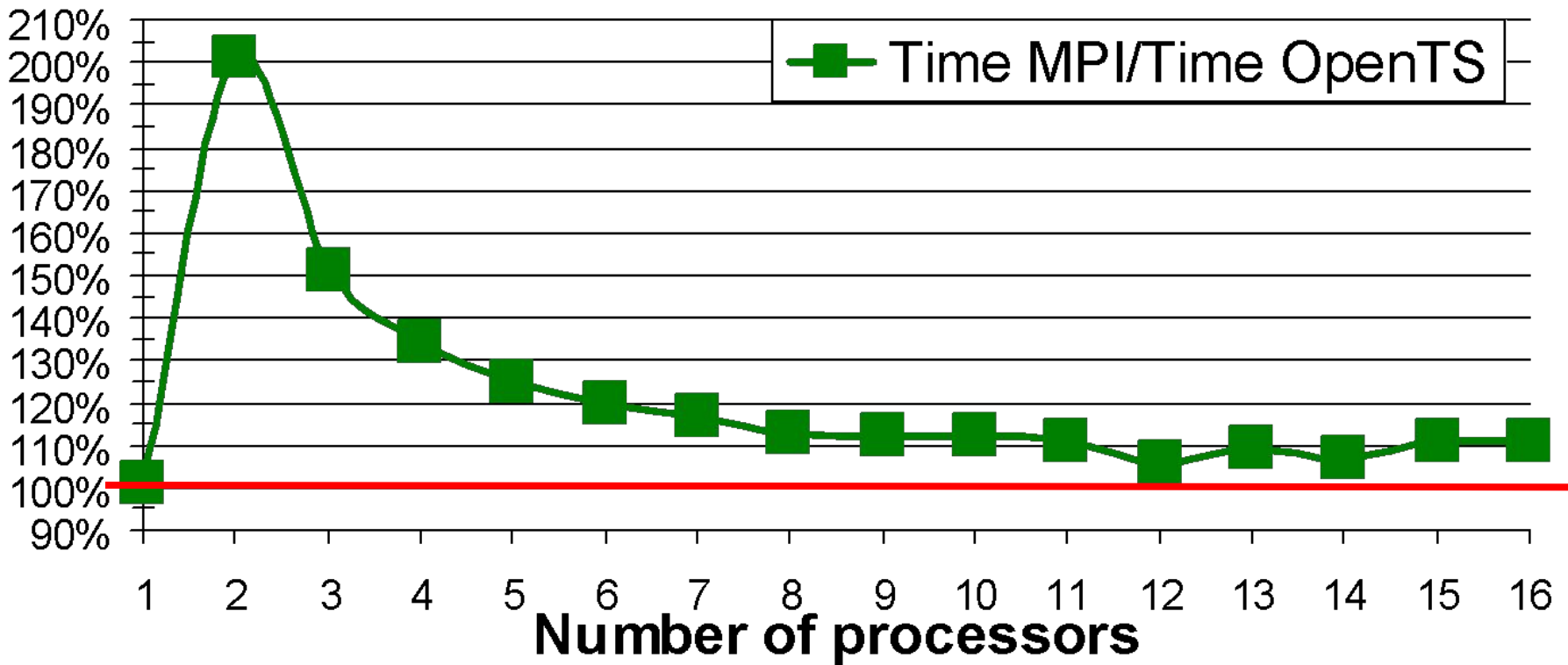
# T-PovRay vs. MPI PovRay: code complexity

| Program                                       | Source code volume |
|---|--------------------|
| MPI modules for PovRay 3.10g                  | 1,500 lines        |
| MPI patch for PovRay 3.50c                    | 3,000 lines        |
| T++ modules (for both versions 3.10g & 3.50c) | 200 lines          |

**~7–15 times**



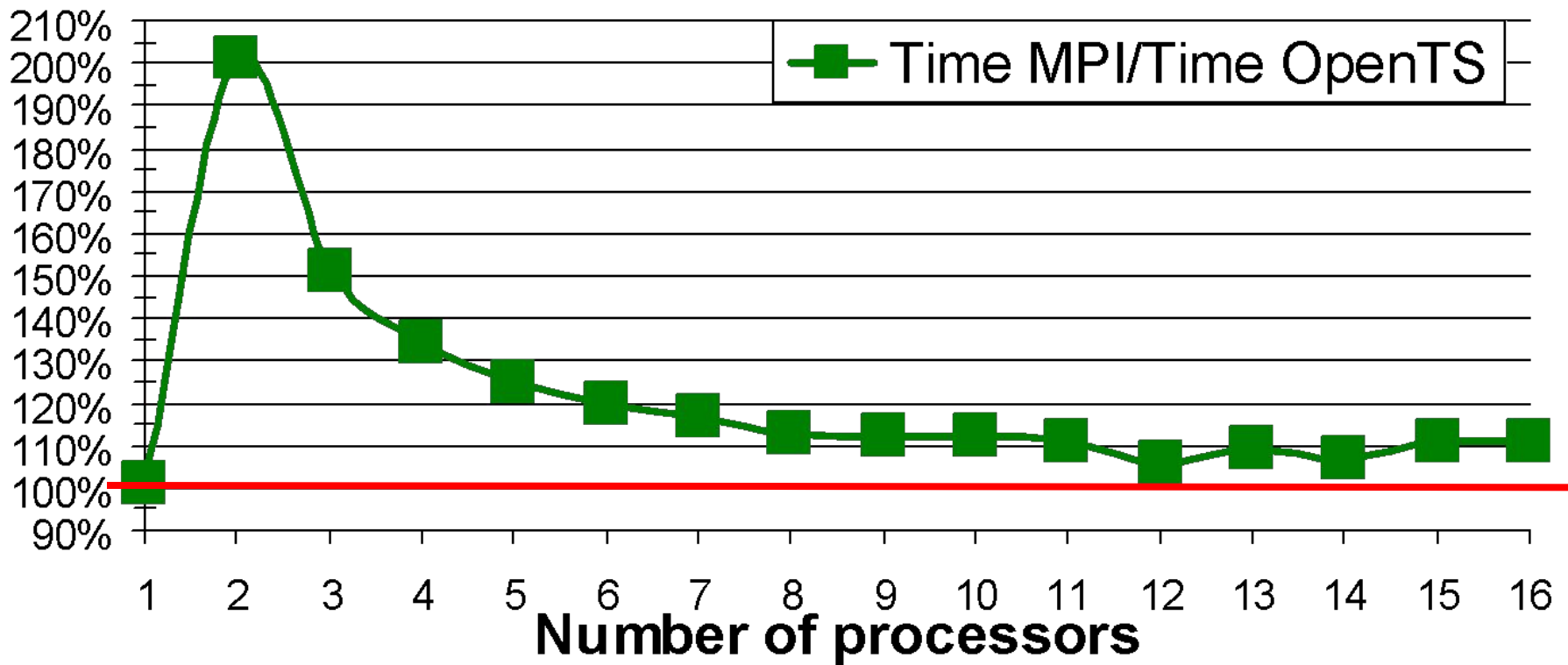
# T-PovRay vs. MPI PovRay: performance



2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,  
GigE, LAM 7.1.1



# T-PovRay vs MPI PovRay: performance

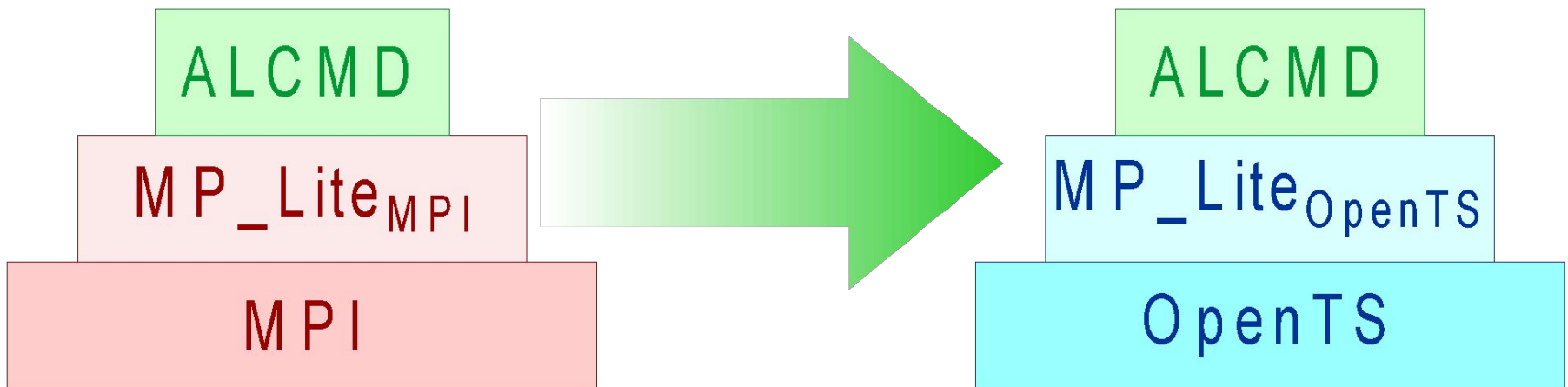


2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,  
GigE, LAM 7.1.1



# ALCMD/MPI vs ALCMD/OpenTS

- ❑ MP\_Lite component of ALCMD rewritten in T++
- ❑ Fortran code is left intact







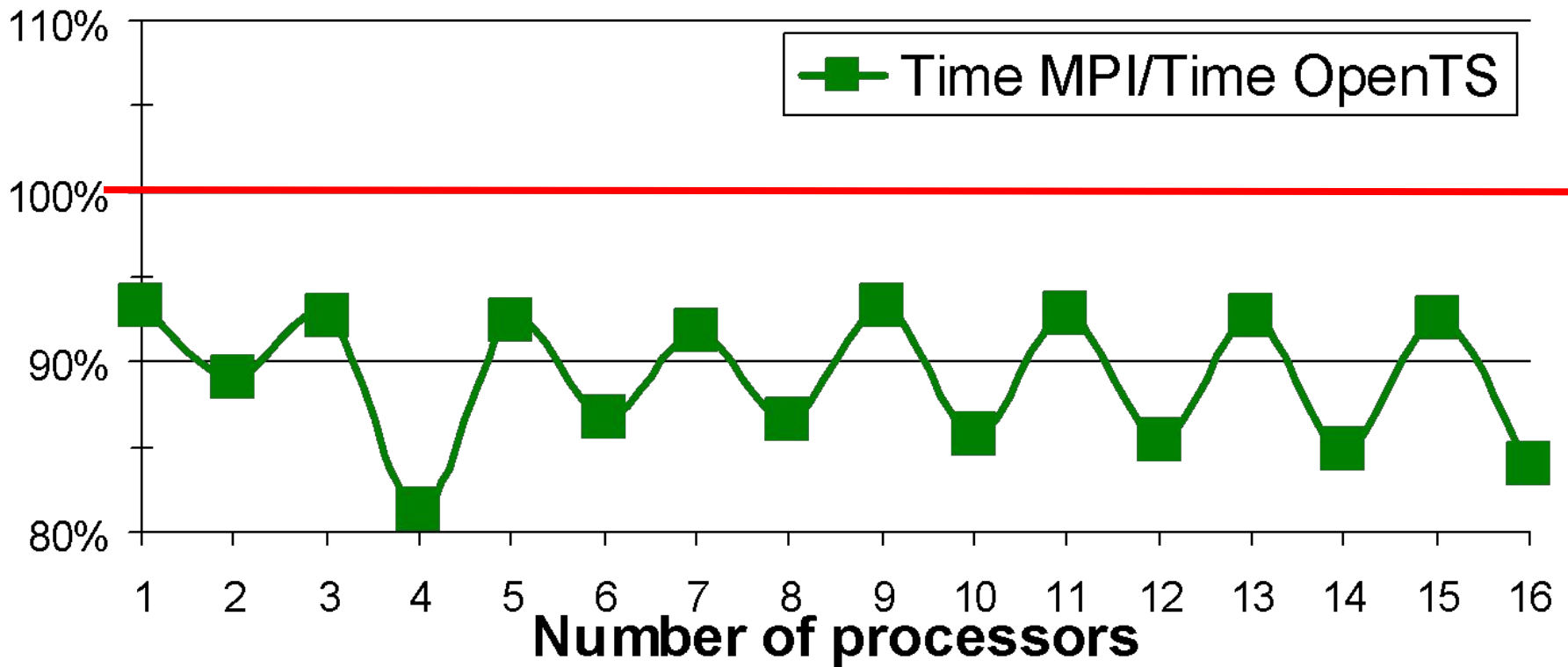
# ALCMD/MPI vs ALCMD/OpenTS : code complexity

| Program                          | Source code volume |
|----------------------------------|--------------------|
| MP_Lite total/MPI                | ~20,000 lines      |
| MP_Lite,ALCMD-related/<br>MPI    | ~3,500 lines       |
| MP_Lite,ALCMD-related/<br>OpenTS | 500 lines          |

**~7 times**



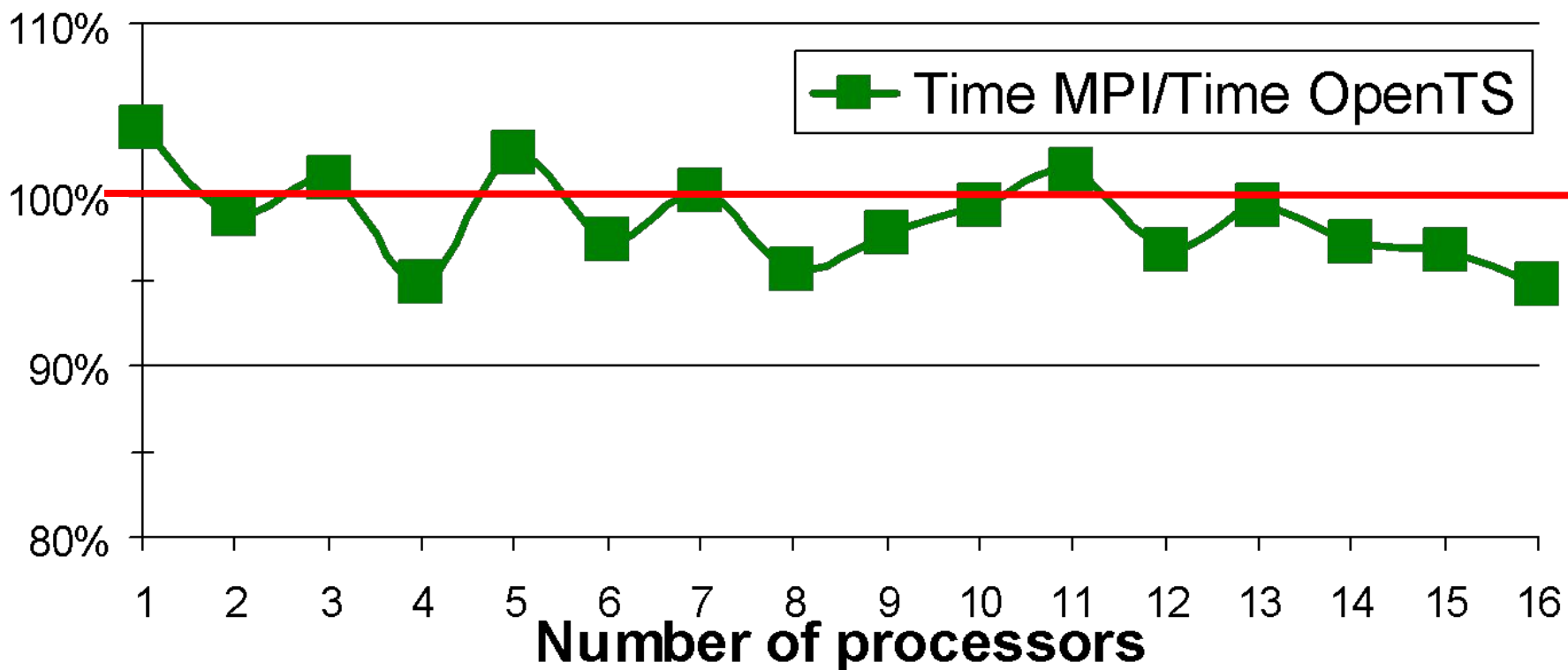
# ALCMD/MPI vs ALCMD/OpenTS: performance



16 dual Athlon 1800, AMD Athlon MP 1800+ RAM 1GB,  
FastEthernet, LAM 7.0.6, Lennard-Jones MD, 512000 atoms



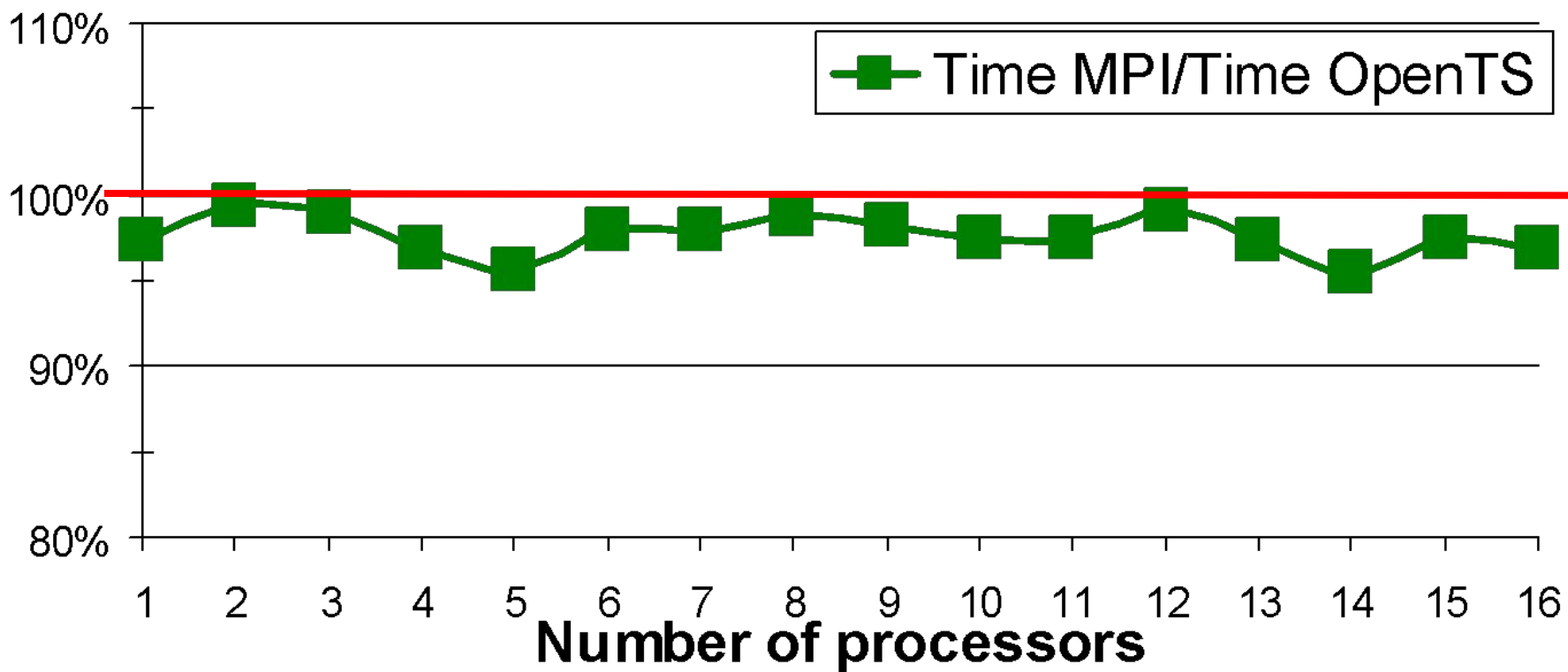
# ALCMD/MPI vs ALCMD/OpenTS: performance



2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,  
GigE, LAM 7.1.1, Lennard-Jones MD, 512000 atoms



# ALCMD/MPI vs ALCMD/OpenTS: performance

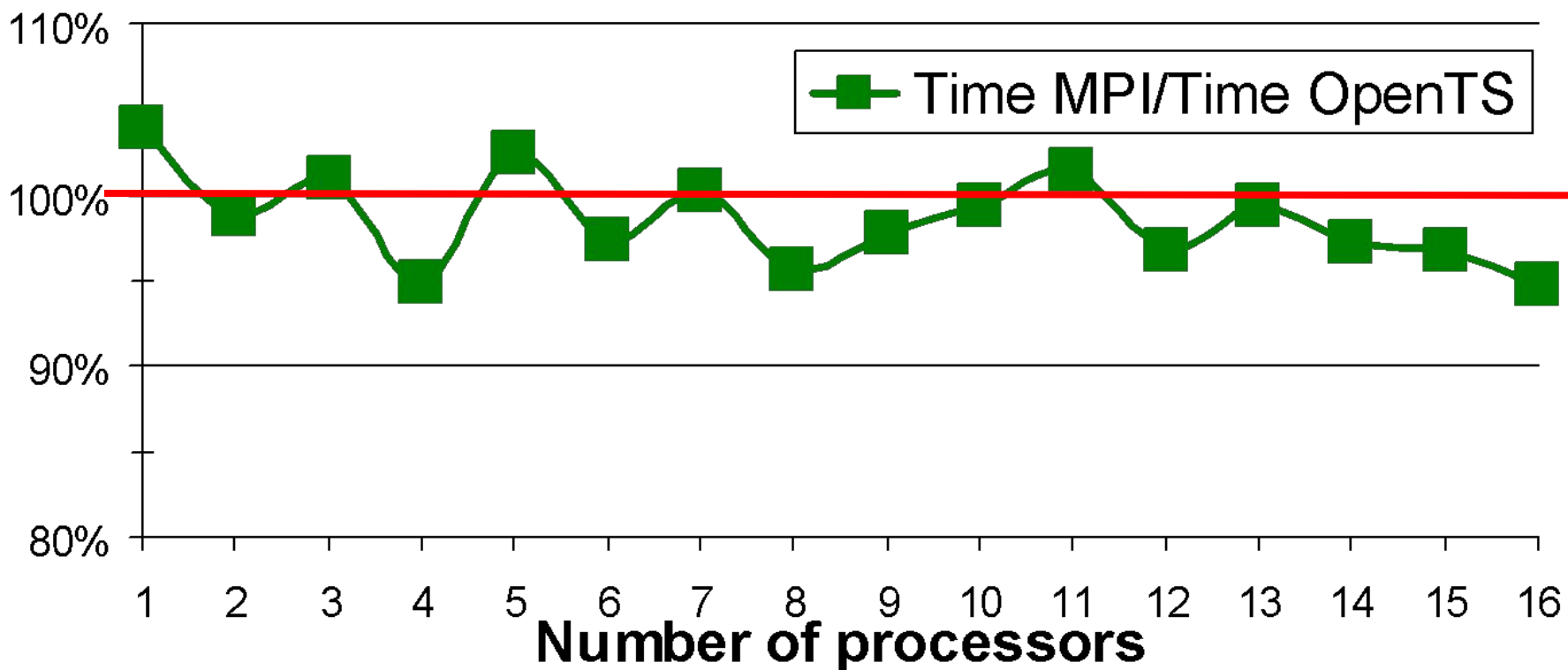


2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,

**InfiniBand**, MVAMPICH 0.9.4, Lennard-Jones MD, 512000 atoms



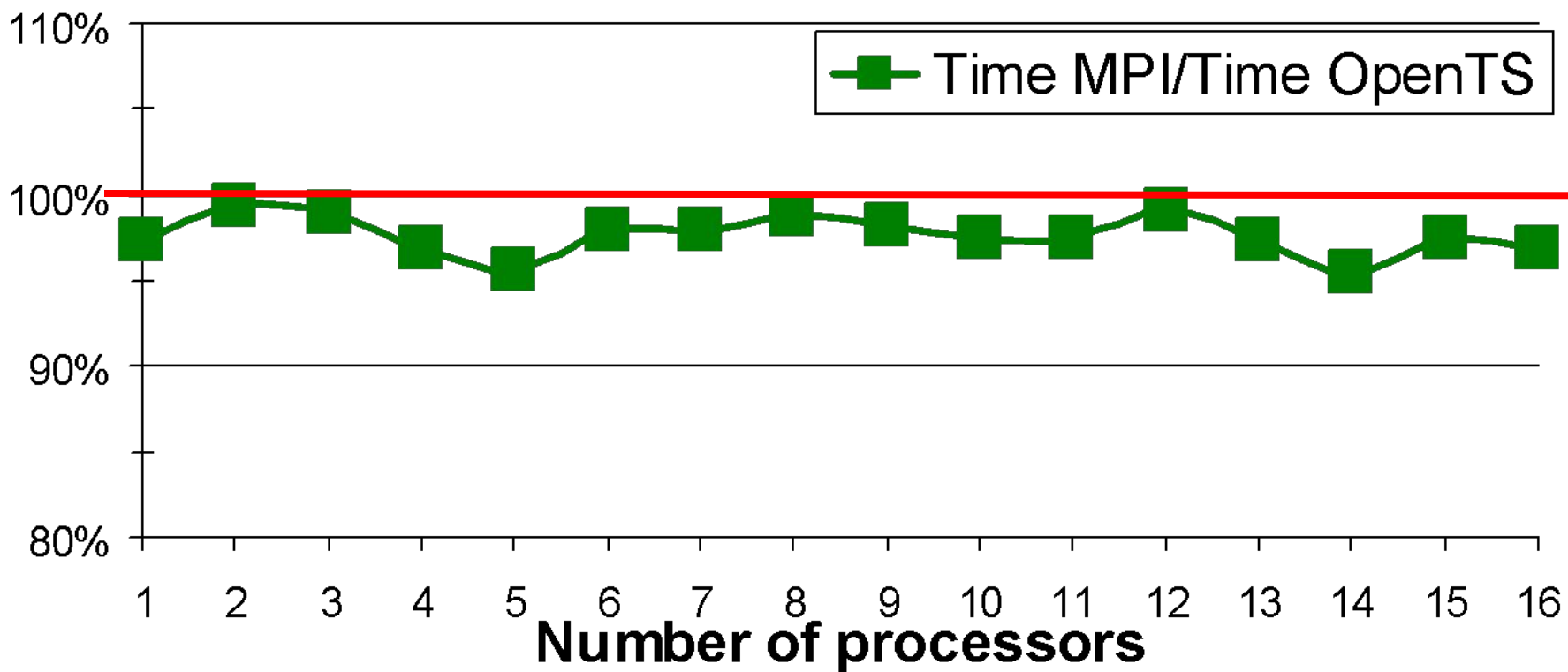
# ALCMD/MPI vs ALCMD/OpenTS: performance



2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,  
GigE, LAM 7.1.1, Lennard-Jones MD, 512000 atoms



# ALCMD/MPI vs ALCMD/OpenTS: performance



2CPUs AMD Opteron 248 2.2 GHz RAM 4GB,

**InfiniBand**, MVAMPICH 0.9.4, Lennard-Jones MD, 512000 atoms



**Program Systems Institute  
Russian Academy of Sciences**

# **Porting OpenTS to MS Windows CCS**





## **2006: contract with Microsoft “Porting OpenTS to Windows Compute Cluster Server”**

- OpenTS@WinCCS
  - ★ inherits all basic features of the original Linux version
  - ★ is available under FreeBSD license
  - ★ does not require any commercial compiler for T-program development — it’s only enough to install VisualC++ 2005 Express Edition (available for free on Microsoft website) and PSDK



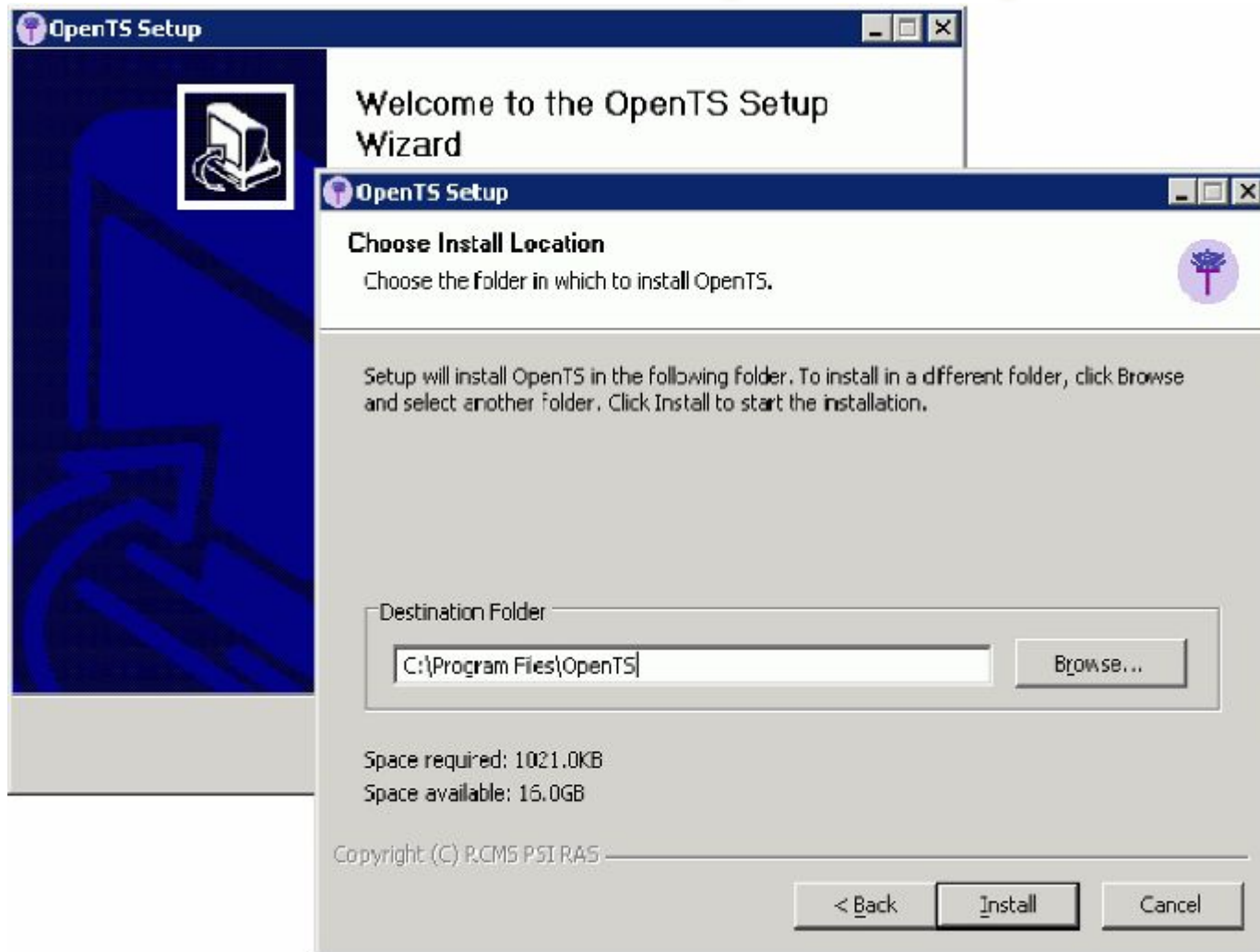


## OpenTS@WinCCS

- ❑ AMD64 and x86 platforms are currently supported
- ❑ Integration into Microsoft Visual Studio 2005
- ❑ Two ways for building T-applications: command line and Visual Studio IDE
- ❑ An installer of OpenTS for Windows XP/2003/WCCS
  - ★ Installation of WCCS SDK (including MS-MPI), if necessary
  - ★ OpenTS self-testing procedure



# Installer of OpenTS for Windows XP/2003/WCCS





Open TS: an advanced tool for parallel and distributed computing.

# OpenTS integration into Microsoft Visual Studio 2005

The screenshot displays the Microsoft Visual Studio 2005 interface. The Solution Explorer on the left shows a project named 'T-Solution' containing a sub-project 'T-App' with 'Header Files', 'Source Files', and a file named 'main.tpp'. The main editor window shows the code for 'main.tpp':

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 tfun int fib (int n)
5 {
6     if (n < 2) return 1;
7     return (fib(n-1) + fib(n-2));
8 }
9 tfun int main (int argc, char* argv[])
10 {
11     int n = atoi(argv[1]);
12     printf("Fib(%d)=%d\n",n,(int)fib(n));
13     return 0;
14 }
```

The Output window at the bottom shows the build process for the 'T-App' project in Release configuration for x64. The output includes the following text:

```
1>----- Build started: Project: T-App, Configuration: Release x64 -----
1>Compiling T-program...
1>"T++->(C++,TSS) Converter v3.0, 2003-2006, PSI RAS, Russia."
1>Converting: occ -v -n -E -STXX -- "d:\VS2005projects\T-Solution\T-App\main.tpp"
1>Load TXX.dll.. Done.
1>[Translate... d:\VS2005projects\T-Solution\T-App\main.tpp into: d:\VS2005projects\T-Solution\T-App\main.occ]
1>[done.]
1>Registered metaclass TXX.
1> tfun int fib:
1> tfun int main:
1>: 2 T-functions
1>Compiling: cl "d:\VS2005projects\T-Solution\T-App\main.occ" /o /Fo"x64\Release\main.obj" /nologo /MT /TP /W0 /EHsc /FItrt
1>main.occ
1>Linking...
1>Build log was saved at "file:///d:/VS2005projects/T-Solution/T-App/x64/Release/BuildLog.htm"
1>T-App - 0 error(s), 0 warning(s)
```

The status bar at the bottom indicates 'Build succeeded' and shows the current cursor position as 'Ln:18 Col:1 Ch:1 INS'.



**Program Systems Institute  
Russian Academy of Sciences**

# **Open TS “Gadgets”**





# Web-services, Live documents

```

tfun int fib (int n) {
    return n < 2 ? n :
        fib(n-1)+fib(n-2);
}

```

## twsgen Perl script

### <operation name="wstfib">

<SOAP:operation style="rpc" soapAction=""/>

### <input>

<SOAP:body use="encoded" namespace="urn:myservice" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>

### </input>

### <output>

<SOAP:body use="encoded" namespace="urn:myservice" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>

### </output>

### </operation>

The screenshot shows two overlapping windows. The top window is 'Microsoft Visual Basic - Workbook.xls - [Module1 (Code)]'. It displays a VBA module with the following code:

```

Function TFib (n As Integer) As Integer
    Dim t As Integer
    debug.print n
    TFib =

```

The 'Tools' menu is open, showing options like 'References...', 'Web Service References...', 'Additional Controls...', 'Macros...', and 'Options...'. The bottom window is 'Microsoft Excel - Workbook'. It shows a spreadsheet with the following data:

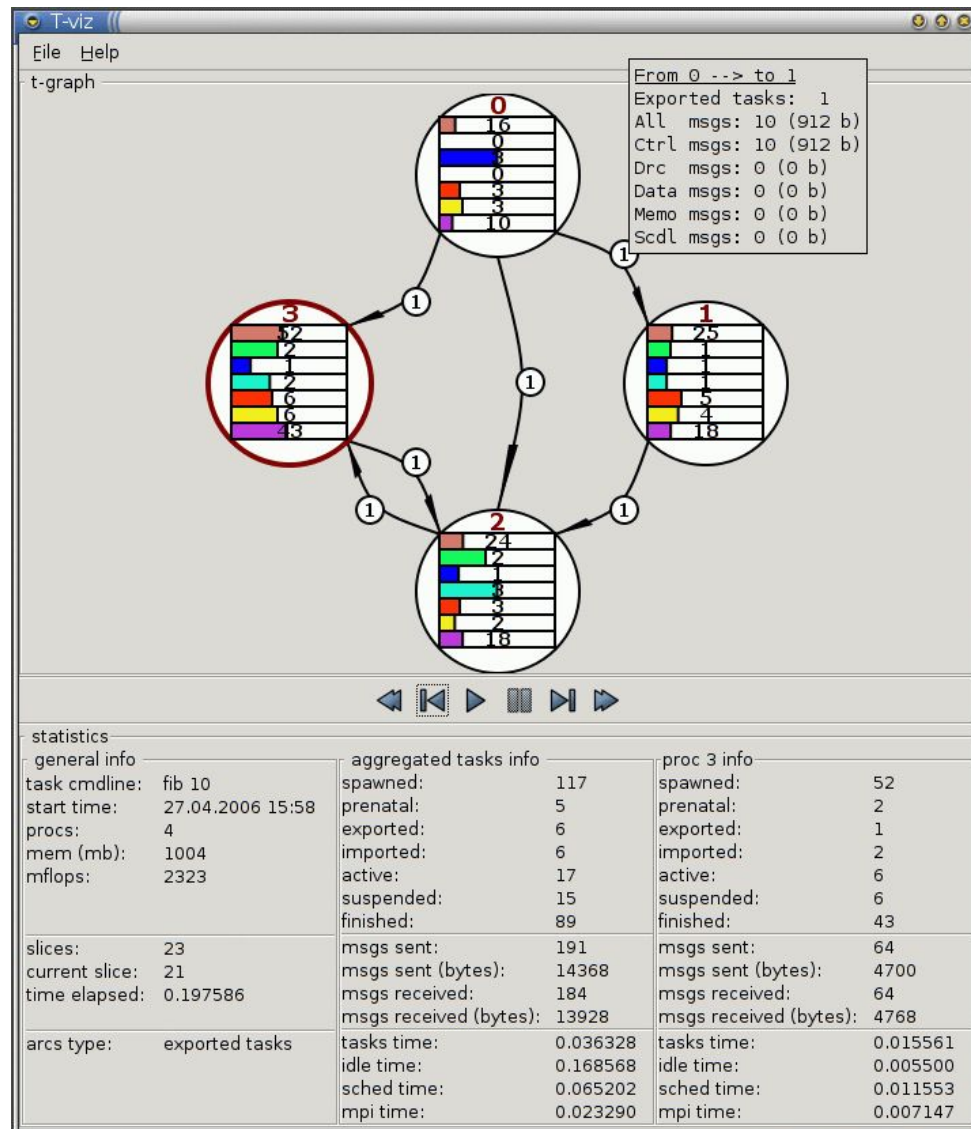
|    | A  | B  | C | D | E | F |
|----|--|----|---|---|---|---|
| 1  |  |    |   |   |   |   |
| 2  | T-Fibonacci web-service (insert arguments into "A" column) |    |   |   |   |   |
| 3  | 1  | 1  |   |   |   |   |
| 4  | 2  | 1  |   |   |   |   |
| 5  | 3  | 2  |   |   |   |   |
| 6  | 4  | 3  |   |   |   |   |
| 7  | 5  | 5  |   |   |   |   |
| 8  | 6  | 8  |   |   |   |   |
| 9  | 7  | 13 |   |   |   |   |
| 10 | 8  | 21 |   |   |   |   |
| 11 | 9  | 34 |   |   |   |   |
| 12 | 10   | 55 |   |   |   |   |
| 13 |  |    |   |   |   |   |

The formula bar shows the formula for cell B3: `=TFib(A3)`.



# Trace visualizer

- ❑ Collect trace of T-program execution
- ❑ Visualize performance metrics of OpenTS runtime





# Fault-tolerance

- Recalculation based fault-tolerance
  - (+) Very simple (in comparison with full transactional model)
  - (+) Efficient (only minimal set of damaged functions are recalculated)
  - (-) Applicable only for functional programs
- Fault-tolerant communications needed (eg.: DMPI v1.0)
- Implemented (experimental version on Linux )



## Some other Gadgets

- ❑ Other T-languages: T-Refal, T-Fortan
- ❑ Memoization
- ❑ Automatically choosing between call-style and fork-style of function invocation
- ❑ Checkpointing
- ❑ Heartbeat mechanism
- ❑ Flavours of data references: “normal”, “glue” and “magnetic” — lazy, eager and ultra-eager (speculative) data transfer





# Full / Empty Bit (FEB) и Т-Система





# FEB

## ★ Модель вычисления:

- «общая память»
- легковесные нити
- FEB — бит синхронизации на каждое слово

## ★ Тонкости: фьючеры

★ **Аппаратная реализация:** Cray XMT, MTA, T3D (2001 и далее)

★ **Программная реализация:** Sandia National Laboratories: Qthreads — [www.cs.sandia.gov/qthreads](http://www.cs.sandia.gov/qthreads)



# Монотонные объекты, как безопасное расширение функциональной модели





## Идеи расширения

- ★ Монотонные объекты — обладают свойством Черча-Россера
- ★ Типичная жизнь монотонного объекта:
  - Создание и инициализация объекта
  - Поток обращений по обновлению («запись») и считыванию текущего состояния («чтение»)
  - Финализация и удаление объекта
- ★ Примеры монотонных объектов (класса сумматоры)
- ★ Примеры приложения с монотонным объектом
- ★ Способы масштабируемой реализации монотонных объектов



**Библиотеки односторонних  
обменов — SHMEM, Gasnet,  
ARMCI**





## Основные черты технологии SHMEM

- ★ «Чужие» данные не обрабатываются на месте, а копируются предварительно туда, где они нужны — как в MPI
- ★ Преобладающий режим копирования – запись, возможно, мелкозернистая
- ★ Синхронизация – барьерная
- ★ Важны
  - высокий message rate (например, для приложений со сложной организацией данных: неструктурных сеток и т.п.)
  - низкая подлинная латентность (для снижения размера зерна параллелизма, независимо от того, сложно или просто организованы данные)
- ★ Однородность доступа не важна
- ★ Message rate можно обеспечить программно, грамотной буферизацией
- ★ Низкую подлинную латентность может дать только аппаратура



## Технология SHMEM

- ★ Рассчитана на полностью однородные многопроцессорные вычислители (общность системы команд, машинного представления чисел, одинаковая операционная система, один и тот же исполняемый файл)
- ★ Программа на С, использующая SHMEM, должна включать файл заголовков `shmem.h`.
- ★ `shmem_init()` — инициализация
- ★ `my_pe()` — собственный номер процесса;
- ★ `num_pes()` — число процессов



## SHMEM: Односторонние обмены

- ★ **put** --- односторонняя запись в чужую память
- ★ **get** --- одностороннее чтение из чужой памяти
- ★ Поддержаны передачи данных разных типов, одного значения, сплошного массива или массива, расположенного в памяти с шагом (например, столбец двумерного массива в C)
- ★ **shmem\_double\_p( addr, value, pe )**  
**shmem\_float\_put( addr, src, len, pe )**
- ★ Можно обмениваться областями памяти статических (но не автоматических!) переменных. И даже поддержан специальный **malloc**





## SHMEM: Операции синхронизации

- ★ Возможность выполнить барьерную синхронизацию всех или лишь указанных процессов.
- ★ При выполнении синхронизации гарантируется, что все выданные до барьера запросы типа **put** будут завершены.
- ★ **shmem\_barrier\_all()**
- ★ **shmem\_barrier( start, stride, size, sync )**
  - **step =  $2^{\text{stride}}$**
  - синхронизируются процессы с номерами **start, start+step, ... start+step\*(size-1)**
  - **sync** — рабочий массив (расписать нулями!) типа **long**, длиной **\_SHMEM\_BARRIER\_SYNC\_SIZE**



## Ожидание переменной

- ★ **shmem\_wait( &var, value )** — на «==»
  - ★ ожидание **&var == value**
  
- ★ **shmem\_int\_wait\_until( &var, SHMEM\_CMP\_GT, value )**
  - ожидание **&var > value**
  
- ★ **shmem\_fence()** — гарантирует, что все выданные ранее из данного процесса запросы типа **put** будут завершены.



**Модели PGAS (Partitioned Global Address Space),  
DSM (Distributed Shared Memory),  
языки Co-Array Fortran, UPC...**





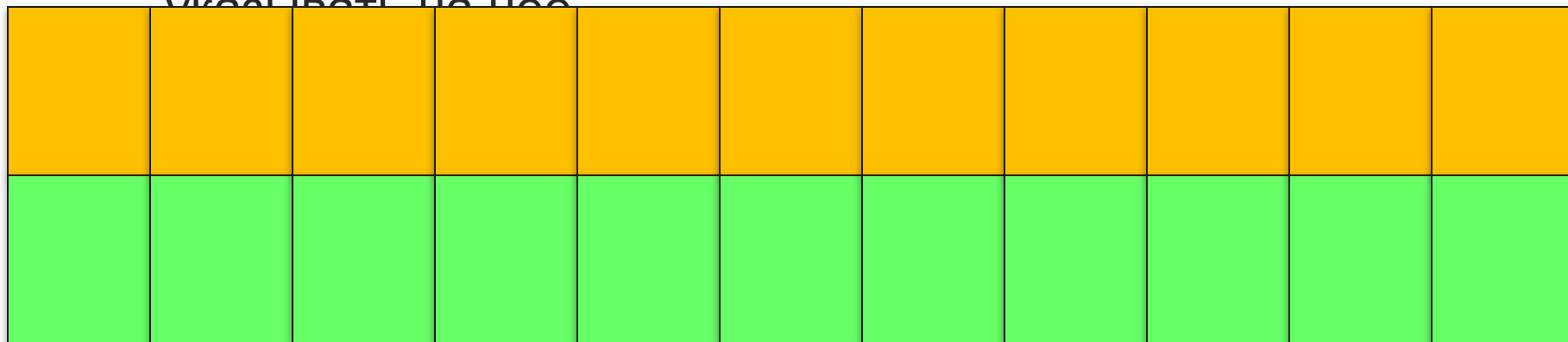
## Модель памяти

### ★ Разделяемая (shared) память

- Любой процесс может использовать ее или указывать на нее

### ★ Приватная память

- Только «локальный» процесс может использовать ее или указывать на нее





## Что такое UPC

- ★ Unified Parallel C
- ★ Расширение ANSI C примитивами задания явного параллелизма
- ★ Основан на «distributed shared memory»
- ★ Основные идеи
  - Сохранить философию C:
  - Программист умен и аккуратен
  - Близость к железу (насколько возможно) для лучшей эффективности (но можно получить и проблемы)
  - Простой и привычный синтаксис



## Модель исполнения

- ★ Несколько процессов (нитей  $0..THREADS-1$ ) работают независимо
- ★ MYTHREAD определяет номер процесса
- ★ THREADS — число процессов
- ★ Когда необходимо для синхронизации используются
  - Барьеры
  - Блокировки
  - Управление поведением памяти



# Пример 1

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS
shared int a[N], b[N], c[N];
void main(){
    int i;
    for(i=0; i<N; i++)
        if (MYTHREAD==i%THREADS)
            c[i]=a[i]+b[i];
}
```



# Вместо заключения







**Спасибо за внимание!**

