ISC-2010, Hamburg, Germany

X-Com:
Distributed Computing Software

Research Computing Center
M.V. Lomonosov Moscow State University

HPC.MSU.RU

X-Com key features (1)

- X-Com takes in account all unique characteristics of distributed computational environments:
  - large scale
    - thousands CPUs
  - geographically distributed resources
    - very high latency in communications
  - variable set of available resources
    - use node on connection
    - don’t fail on node disconnection
  - heterogeneous resources:
    - operating system (MS Windows, Linux, …)
    - CPU (x86, x86_64, ia64, …)

X-Com key features (2)

- Lightweight portable toolkit:
  - easy to install
    - Perl-based software
    - doesn’t need administrator/root access
  - easy applications adaptation
    - maybe several Perl strings must be written
  - simultaneous work of different platforms
    - joining Windows and Linux machines
  - distributed freely
    - BSD license

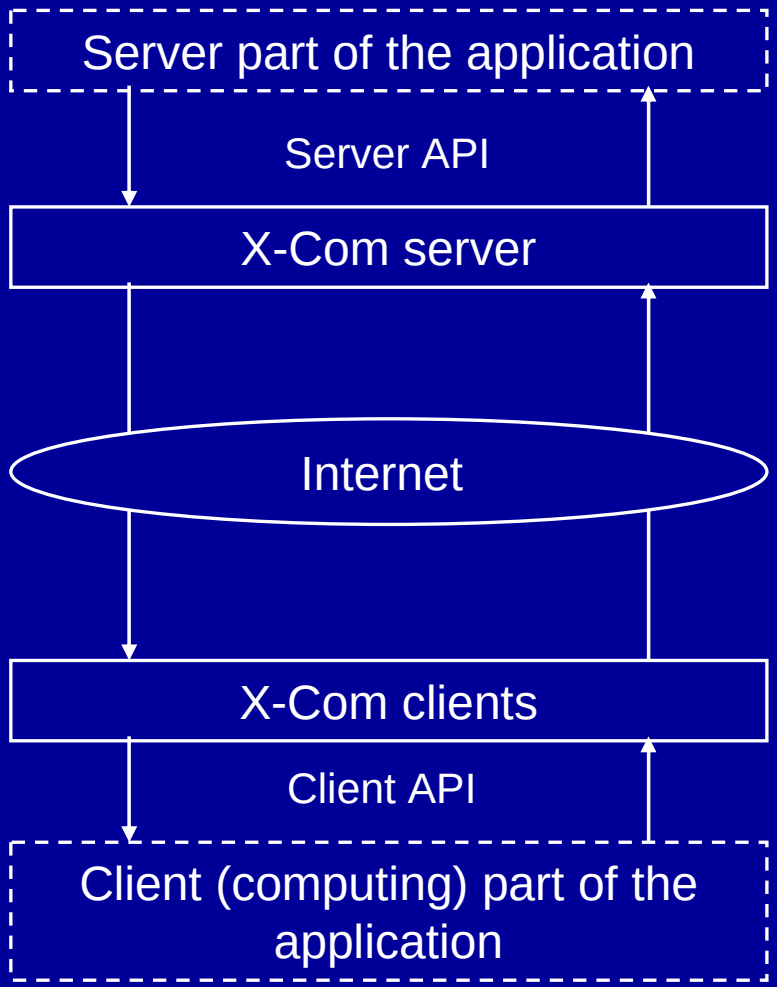
X-Com basic requirements

- Linux / Windows
- Perl 5.8.0+
  - for Windows:
    - Strawberry Perl (preferred)
    - ActivePerl
- Open TCP port for communications
  - if firewall is used
- Disabled CPU time limits
  - if used on head machine of computing cluster

Structure of application in X-Com distributed environment

- Application:
  - ability to split a task into a number of independent subtasks (portions)
  - more computing, less communicating
- X-Com – clients & server
  - server is responsible for splitting tasks into portions and joining results
  - clients receive data from server, perform computations, send results back

General idea of X-Com computing



X-Com server APIs

- Files API
  - for quite simple applications
  - only need to specify input/output paths and a file list
- Perl API
  - for more complicated applications
  - need to create 6 functions in a Perl module

X-Com server APIs: Files

- Options of the server settings file (ini-file):
  - taskList
    - input files list
  - taskIn
    - path where input files are stored
  - taskOut
    - output path

X-Com server APIs: Perl

- Functions to be write in a task server module:
  - initialize ($taskArg)
    - performs starting actions, e.g. arguments checking
  - getFirstPortionNumber ()
    - returns number of the 1st portion (usually returns 1)
  - getLastPortionNumber ()
    - returns number of the last portion (if we know total number of portions)
  - isFinished ()
    - checks finishing conditions and returns true if the work is over (if we don’t know total portions number)
  - getPortion ($N)
    - returns content of Nth portions
  - addPortion ($N, $data)
    - processes the result of Nth portions which is stored in $data
  - finalize ()
    - makes any final actions

X-Com client APIs

- Processes API
  - only need to specify command lines for initializing and portion processing
- Perl API
  - need to create 2 Perl functions

X-Com client APIs: Processes

- Options of the server settings file (ini-file):
  - xCliInitCmd
    - command line which will be run once before portions processing
  - xCliPortionCmd
    - command line used for processing each portion
  - xCliPortionIn
    - file name in which incoming portions will be stored
  - xCliPortionOut
    - file name from which resulting data well be taken

X-Com client APIs: Perl

Functions to be write in a client module:

- gcprepare ($addarg)
  - function called once before any portion-processing actions, i.e. client part initializing
    - $addarg is an additional parameter that can be passed to the X-Com client from command line
- gctask ($task, $taskarg, $portion, $din, $dout, $addarg)
  - function called for every portion
    - $task - task name
    - $taskarg - task arguments
    - $portion - portion number
    - $din - file with portion content
    - $dout - file from which results well be read

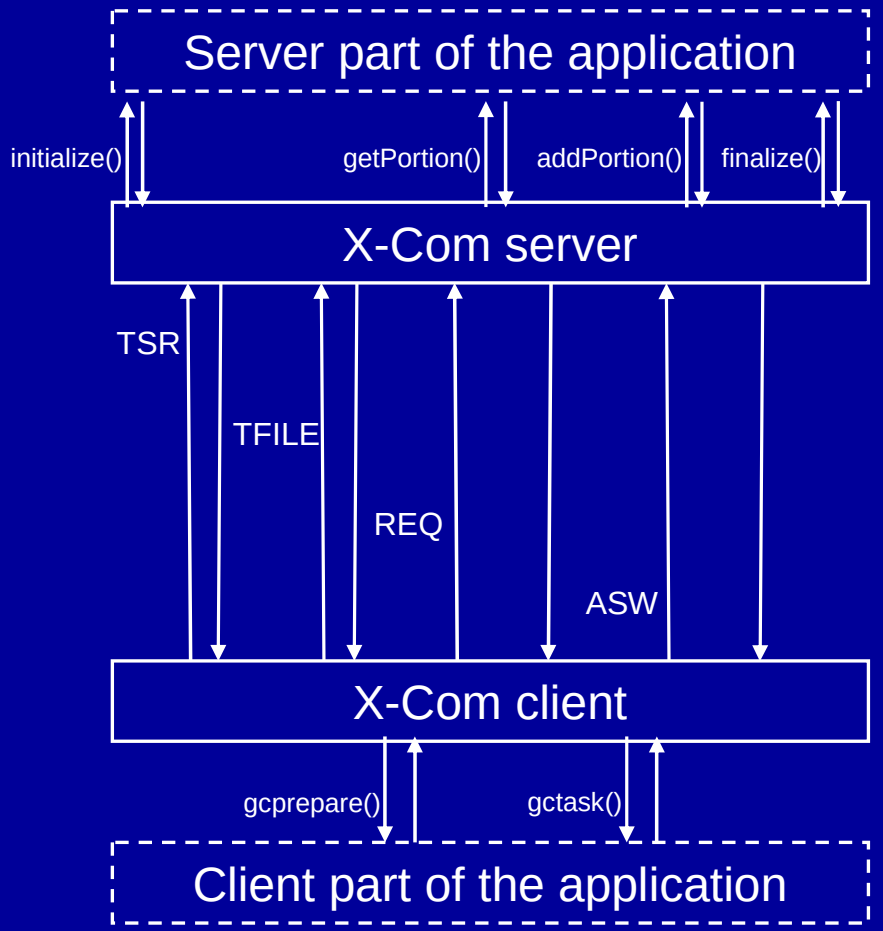
X-Com computing sequence using Perl API on both server and clients

1. Task initializing on server side

2. Task description request
3. Request for required files
4. Task initializing on clients

5. Portion request
6. Computations
7. Result sending

8. Finalizing



X-Com files structure at a glance

- **gserv** **- X-Com server**
  - **XServ.pl** - executable of the server
  - ***.pm** - server components
  - ***.ini** - ini-files of server
  - **logs** - log-files saving path
- **tasks** **- applications**
  - **Pi** - Pi calculation application (sample task)
    - **Pi.pm** - server part interface of Pi app.
    - **gctask** - client part interface of Pi app.
    - ***.tar.gz** - packed client parts
- **gcli** **- X-Com client**
  - **client.pl** - executable of the client
  - ***.pm** - client components
- **utils** **- utilities**
  - **XLogProc.pl** - log-files analyzing tool

Simple example: distributed grep. X-Com server settings & running

```
# dgrep.ini

# Server info

prSock          = /tmp/xcom.sock        # internal UNIX socket, need for server
ssHost          = newserv.srcc.msu.ru   # external TCP interface for clients – host…
ssPort          = 65002                 # … and port

# Task info

taskName        = dgrep                   # task name
taskAPI         = Files                   # task API
taskArgs        = 212.192.244             # arguments (a substring to search in files)
taskList        = /tmp/alogs.in/in.txt    # files list
taskIn          = /tmp/alogs.in           # input path (Apache log-files, by day)
taskOut         = /tmp/alogs.out          # output path
```

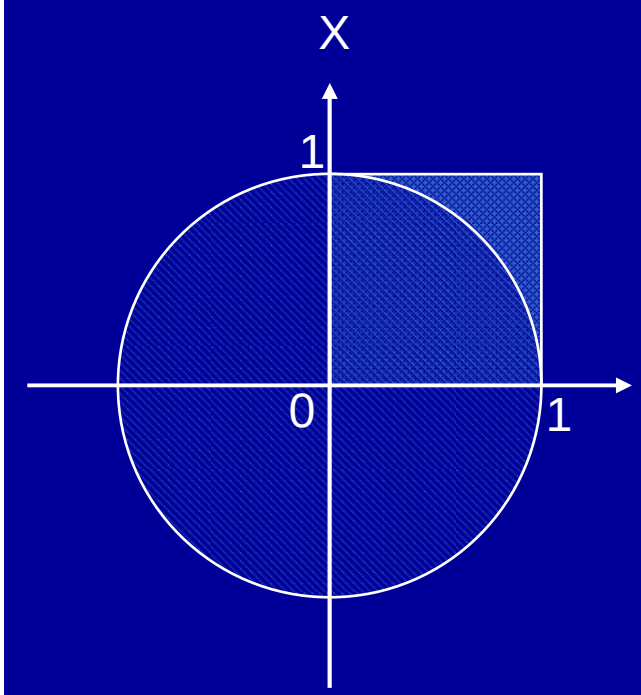
```
$ ./XServ.pl dgrep.ini
```

Simple example: distributed grep. Client part functions & running

```
sub gcprepare {
  return 1;
}

sub gctask {
  my ($task, $taskarg, $portion, $din, $dout) = @_;
  my ($file, $pattern) = split (' ', $taskarg);
  `/bin/grep '$pattern' $din > $dout`;
  return 1;
}

1;
```

```
$ ./client.pl –s http://newserv.srcc.msu.ru:65002
```

Another example: Pi calculation using Monte Carlo method



Let $X_n = (x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)$ are n pairs of independent random values with normal distribution

Let K_{Xn} is a number of pairs for which $(x_i^2+y_i^2) < 1$

Than $4K_{Xn}/n \rightarrow \pi,\ n\rightarrow\infty$

Accuracy with quite large n: up to i-th sign after dot, where $i = [-\log_{10}(2/(3n)^{½})]-1$

Pi calculating: server part functions (1)

~/xcom/tasks/Pi/Pi.pm

```
sub initialize {
   ($portionsCount,$expo) = split (/ /, $_[0])
        or die ("Pi error: incorrect arguments!");
   $totalSum = 0;
   $donePortions = 0;
   $outCSV = "../tasks/Pi/out/pi_esteem_$portionsCount"."_$expo.csv";
   $outRes = "../tasks/Pi/out/pi_$portionsCount"."_$expo.txt";
   open E, "> $outCSV"
        or die ("Pi error: can't write to $outCSV!");
   print E 'Portions;Drops;EstimatedPi;RealPi;Error'."\n";
   close E;
   print STDERR "Pi initialized: $portionsCount portions,
        ".(10**$expo)." (10**$expo) drops in portion\n";
   return 1;
}

sub getFirstPortionNumber {
   return 1;
}

sub getLastPortionNumber {
   return $portionsCount;
}
```

Pi calculating: server part functions (2)

~/xcom/tasks/Pi/Pi.pm

```
sub getPortion {
   my ($portion) = @_;
   return $portion;
}

sub addPortion {
   my ($n, $portionSum) = ($_[0], $_[1]);
   $totalSum = $totalSum + $portionSum;
   $donePortions += 1;
   my $dropsCount = $donePortions * (10 ** $expo);
   $thePI = $totalSum / $dropsCount;
   open E, ">> $outCSV"
        or die ("Pi error: can't write to $outCSV!");
   my $lc_numeric = setlocale(LC_NUMERIC);
   setlocale(LC_NUMERIC,'ru_RU');
   print E "$donePortions;$dropsCount;$thePI;$PI;".
        ($thePI-$PI)."\n";
   setlocale(LC_NUMERIC,$lc_numeric);
   close E;
}
```

Pi calculating: server part functions (3)

~/xcom/tasks/Pi/Pi.pm

```
sub isFinished {
   return 0;
}

sub finalize {
   open F, "> $outRes"
        or die ("Pi error: can't write to $outRes");
   print F $thePI."\n";
   close F;
   return 1;
}
```

Pi calculating: client part functions
~/xcom/tasks/Pi/gctask

```
sub gcprepare {
   return 1;
}

sub gctask {
   my ($task,$taskarg,$portion,$din,$dout) = @_;
   my ($n,$expo) = split (/ /, $taskarg);
   my $cmd = "Pi $expo";
   $cmd = "./$cmd" if ($^O ne 'MSWin32');
   my $s = readpipe($cmd);
   if ($s==-1) {
        return 0;
   }
   open (F,"> $dout");
   print F "$s";
   close (F);
   return 1;
}
```

Pi calculating: X-Com server settings

~/xcom/gserv/Pi.ini

```
# Pi.ini

# Server info

prHost      = 212.192.244.31   # host-port pair for internal server purposes…
prPort      = 65000            # … that can be used instead of UNIX socket
ssHost      = 212.192.244.31   # external (client) interface: host…
ssPort      = 65001            # … and port

# Task info

taskName    = Pi
taskAPI     = Perl
taskArgs    = 100 9            # 100 portions, 10^9 random drops in each
```

X-Com server console: starting ~/xcom/gserv

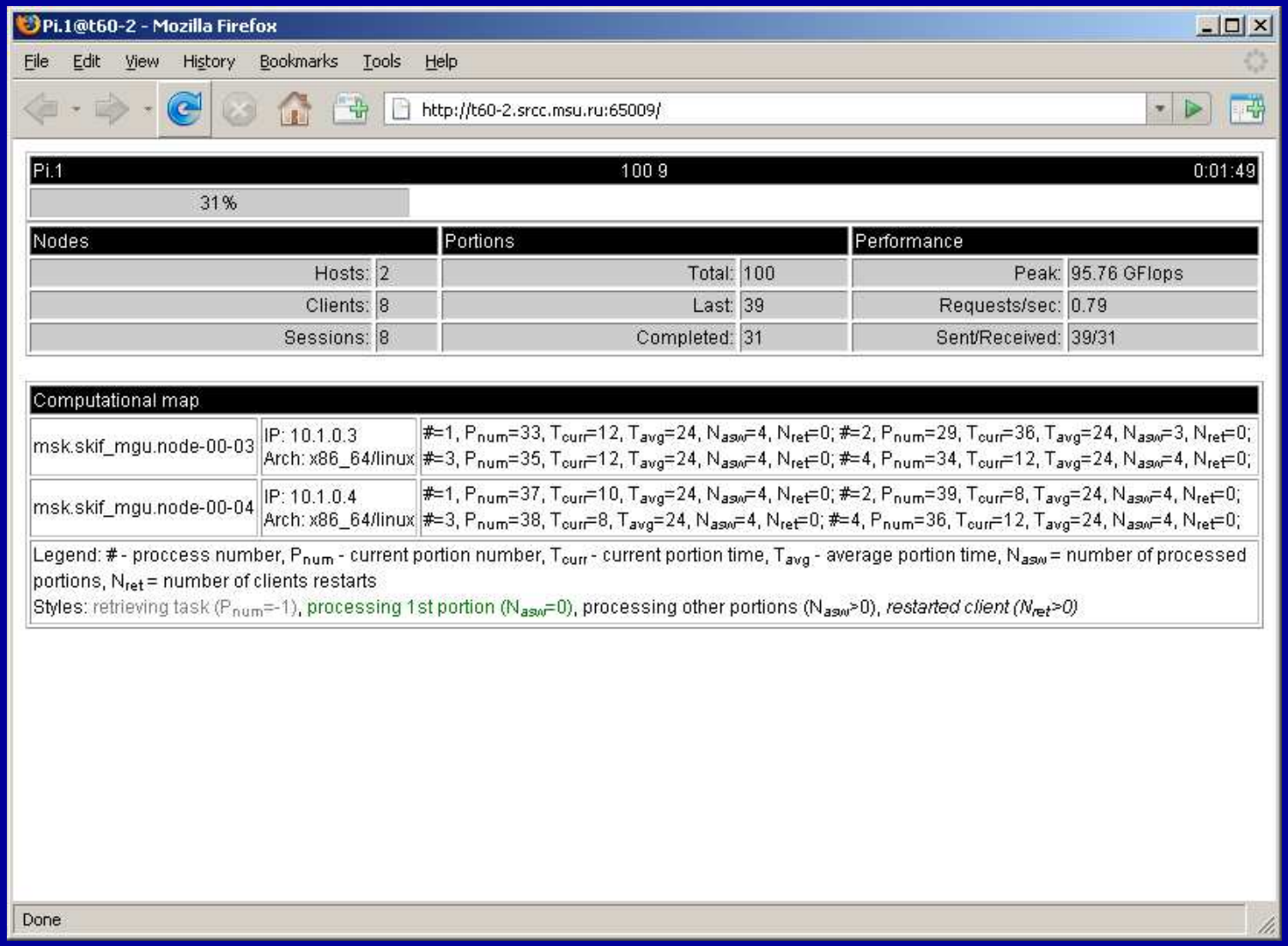
```
$ ./XServ.pl Pi.ini

XServ: starting
 Processor:              212.192.244.31:65000
 Subservers:             212.192.244.31:65001
 Download prefix:        {http://212.192.244.31:65001/}
 Task API:               Perl
 Task name:              Pi
 Task ID:                1
 Task arguments:         100 9

Pi initialized: 100 portions, 1000000000 (10**9) drops in portion
XServProcessor: TestPi initialized (1 -> 100)
```

X-Com server console: basic monitoring of computing process

```
P> msk.skif_mgu.node-00-04.3: TSR
P> msk.skif_mgu.node-00-04.1: TSR
P> msk.skif_mgu.node-00-03.4: TSR
P> msk.skif_mgu.node-00-04.4: TSR
P> msk.skif_mgu.node-00-04.2: TSR
P> msk.skif_mgu.node-00-03.2: TSR
P> msk.skif_mgu.node-00-03.1: TSR
P> msk.skif_mgu.node-00-03.3: TSR
P> msk.skif_mgu.node-00-03.4: REQ 1
P> msk.skif_mgu.node-00-03.1: REQ 2
P> msk.skif_mgu.node-00-03.2: REQ 3
P> msk.skif_mgu.node-00-04.2: REQ 4
P> msk.skif_mgu.node-00-04.4: REQ 5
P> msk.skif_mgu.node-00-04.1: REQ 6
P> msk.skif_mgu.node-00-03.3: REQ 7
P> msk.skif_mgu.node-00-04.3: REQ 8
P> msk.skif_mgu.node-00-03.4: ASW 1
P> msk.skif_mgu.node-00-03.1: ASW 2
P> msk.skif_mgu.node-00-03.4: REQ 9
P> msk.skif_mgu.node-00-03.1: REQ 10
…
```

Web-based monitoring

http://<host:port>/

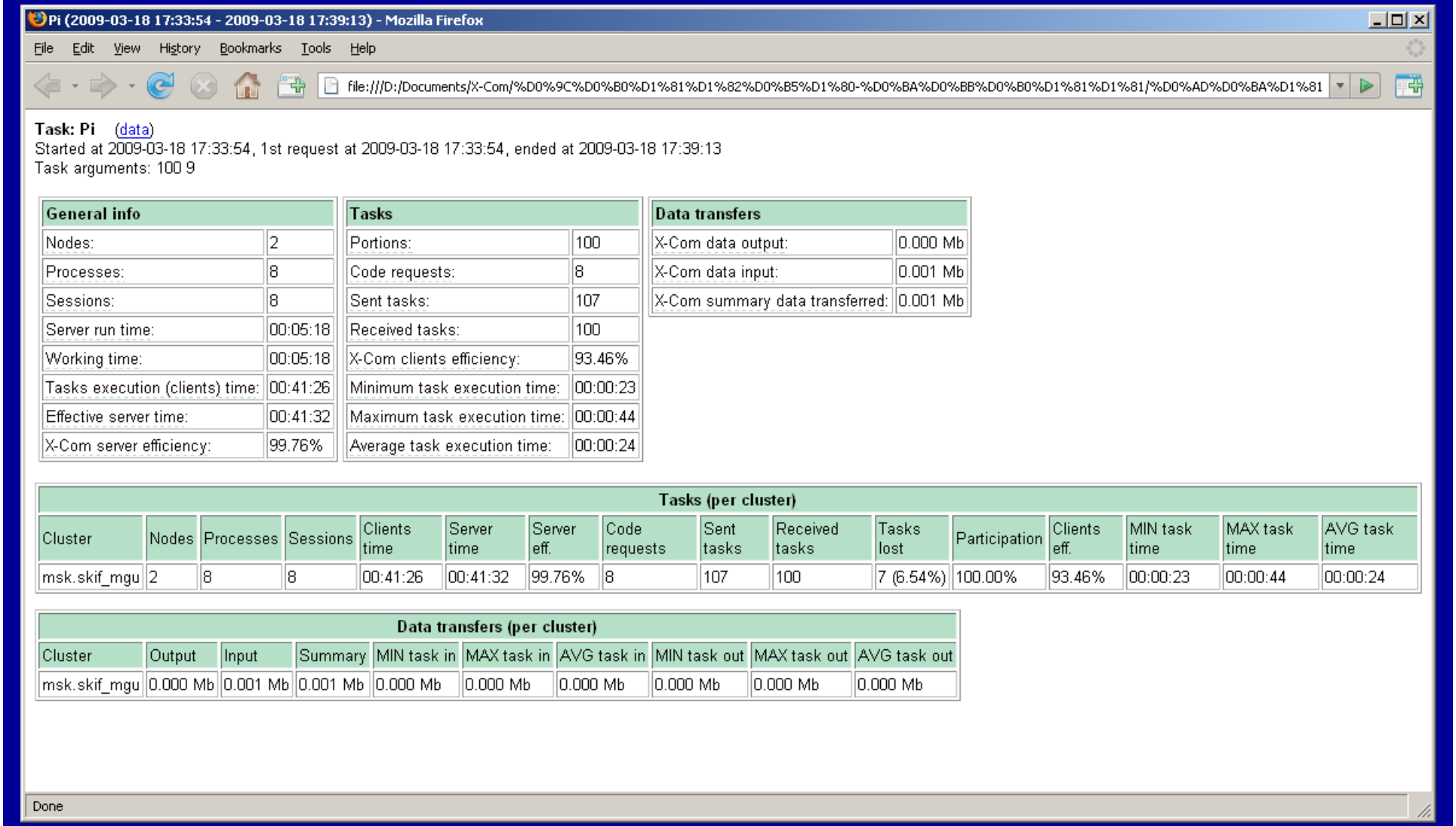


X-Com server console: finishing

```
…
P> msk.skif_mgu.node-33-08.1: ASW 96
P> msk.skif_mgu.node-33-08.1: REQ 97
P> msk.skif_mgu.node-33-03.1: ASW 97
P> msk.skif_mgu.node-33-03.1: REQ 98
P> msk.skif_mgu.node-33-05.1: ASW 98
P> msk.skif_mgu.node-33-05.1: REQ 99
P> msk.skif_mgu.node-33-07.1: ASW 99
P> msk.skif_mgu.node-33-07.1: REQ 100
P> msk.skif_mgu.node-33-06.1: ASW 95
P> msk.skif_mgu.node-33-06.1: REQ 100
P> msk.skif_mgu.node-33-02.1: ASW 100
P> Finishing communications...
XServSubserver: cannot connect to 212.192.244.31:65000, retrying (1)...
XServSubserver: cannot connect to 212.192.244.31:65000, retrying (2)...
Killed
$
```

Analysis of the finished calculations

~/xcom/utils/XLogProc.pl



Pi (2009-03-18 17:33:54 - 2009-03-18 17:39:13) - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///D:/Documents/X-Com/%D0%9C%D0%B0%D1%81%D1%82%D0%B5%D1%80-%D0%BA%D0%BB%D0%B0%D1%81%D1%81/%D0%AD%D0%BA%D1%81

Task: Pi (data)
Started at 2009-03-18 17:33:54, 1st request at 2009-03-18 17:33:54, ended at 2009-03-18 17:39:13
Task arguments: 100 9

General info	
Nodes:	2
Processes:	8
Sessions:	8
Server run time:	00:05:18
Working time:	00:05:18
Tasks execution (clients) time:	00:41:26
Effective server time:	00:41:32
X-Com server efficiency:	99.76%

Tasks	
Portions:	100
Code requests:	8
Sent tasks:	107
Received tasks:	100
X-Com clients efficiency:	93.46%
Minimum task execution time:	00:00:23
Maximum task execution time:	00:00:44
Average task execution time:	00:00:24

Data transfers	
X-Com data output:	0.000 Mb
X-Com data input:	0.001 Mb
X-Com summary data transferred:	0.001 Mb

Tasks (per cluster)

Cluster	Nodes	Processes	Sessions	Clients time	Server time	Server eff.	Code requests	Sent tasks	Received tasks	Tasks lost	Participation	Clients eff.	MIN task time	MAX task time	AVG task time
msk.skif_mgu	2	8	8	00:41:26	00:41:32	99.76%	8	107	100	7 (6.54%)	100.00%	93.46%	00:00:23	00:00:44	00:00:24

Data transfers (per cluster)

Cluster	Output	Input	Summary	MIN task in	MAX task in	AVG task in	MIN task out	MAX task out	AVG task out
msk.skif_mgu	0.000 Mb	0.001 Mb	0.001 Mb	0.000 Mb	0.000 Mb	0.000 Mb	0.000 Mb	0.000 Mb	0.000 Mb

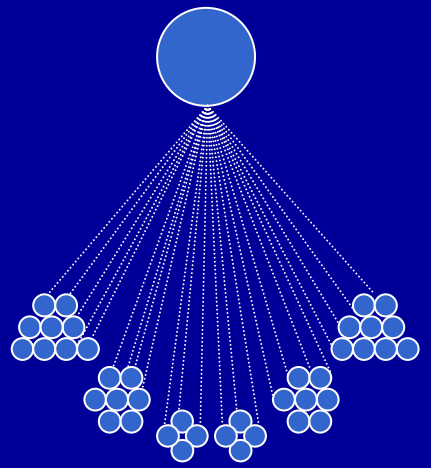
Done

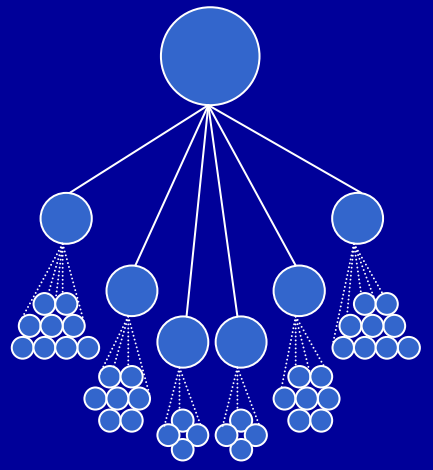
X-Com additional features: model (testing) task

- The task aimed to model real applications for preliminary assessment of environment characteristics and application behavior
- Parameters:
  - size of input/output data
  - time of portion processing
  - computational modules

X-Com additional features: proxy (buffering) servers

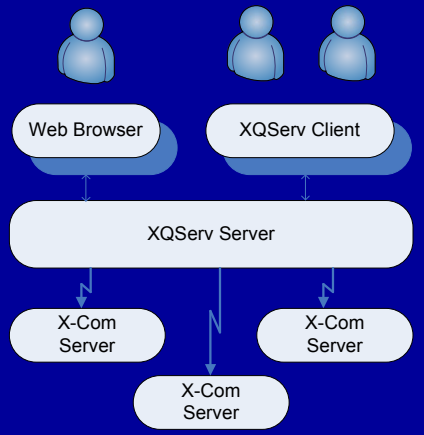
- Unlimited levels of hierarchy
- Data (portions and files) buffering
  - less network connections
  - less traffic consumption
- Optimization of server load

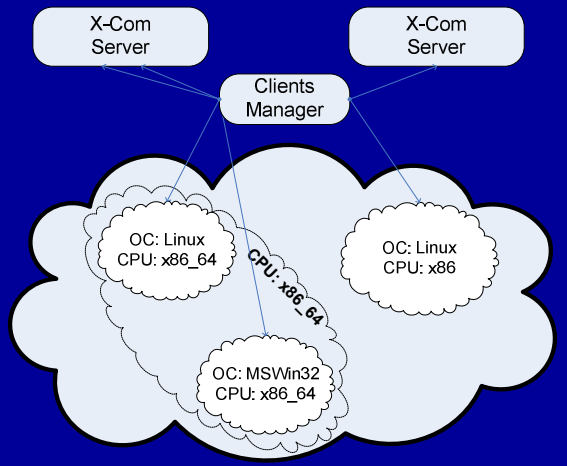




X-Com additional features:
XQServ - task management subsystem

- High-level user interface
  - almost as common batch system
- Multiuser and multitasking
- Tasks scheduling:
  - consecutive queue
  - parallel running
  - using specified task conditions:
    - CPU type, frequency, performance
    - OS type
    - RAM size

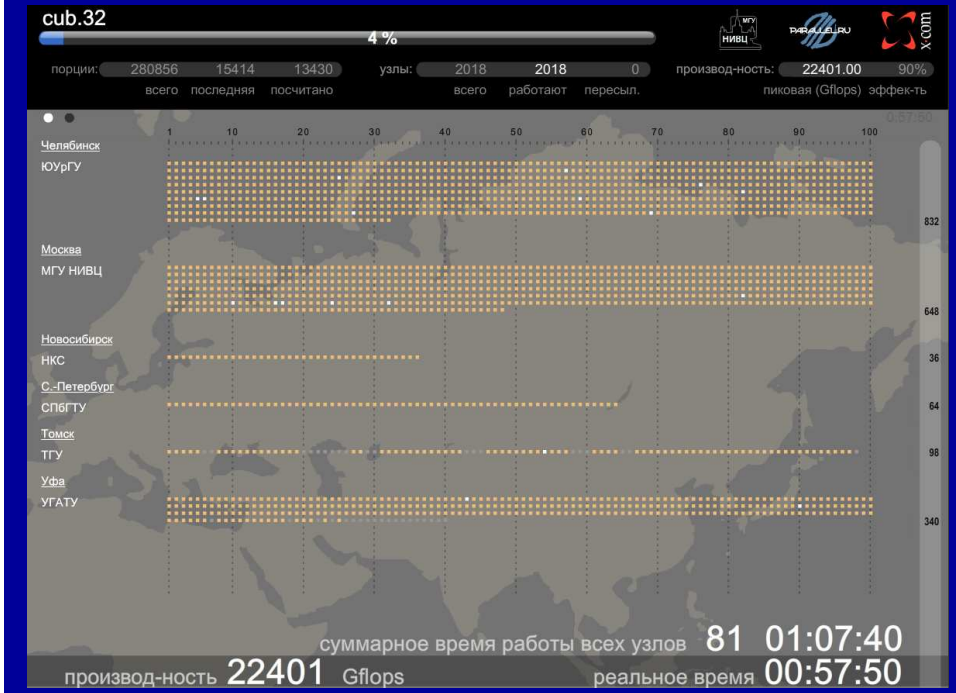


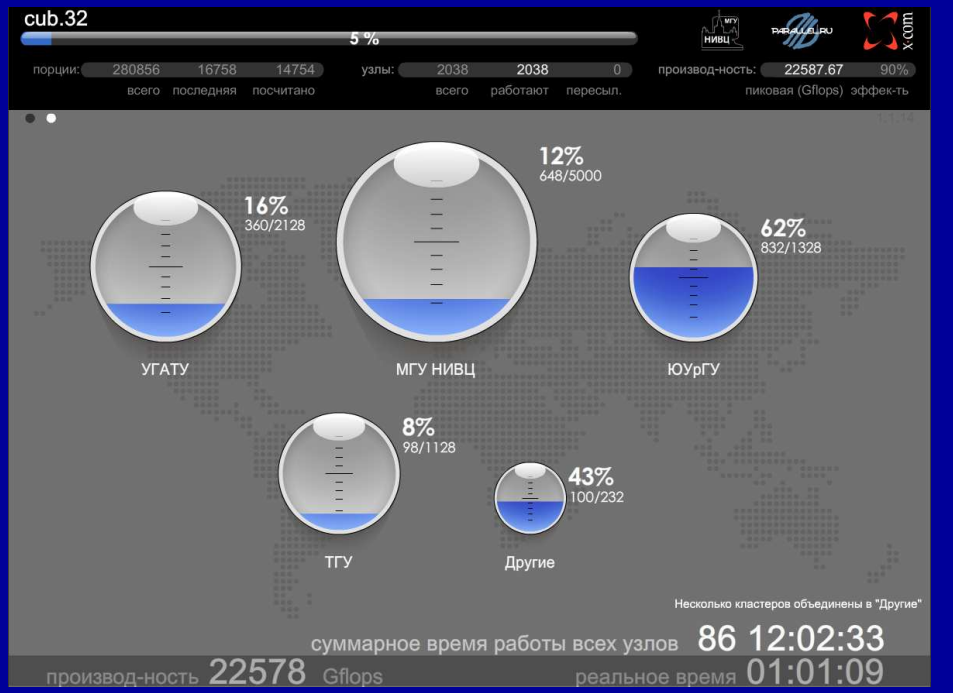


X-Com additional features: working on computing clusters

- Exclusive nodes using
- Using nodes idle time
- Using batch systems:
  - Torque
  - LoadLeveler
  - Cleo
  - *Unicore*

X-Com additional features: Flash-based computing visualization





Real distributed computing practice (1)

Computing of matrix coefficients for electromagnetic field diffraction on homogeneous dielectric free shape objects (together with Penza State University)

Computing resources – 6 HPC sites:

- Moscow, Chelyabinsk, Ufa, Sankt-Petersburg, Novosibirsk, Tomsk
  - Linux OS
  - Intel Xeon, Intel Itanium, AMD Opteron CPUs
  - exclusive using of nodes
- 2199 cores
- 24.5 TFlops

Task splitting:

- 280525 portions, about 5 minutes processing each

Time:

- total – 11 hours 46 minutes
- CPU – 2.7 years

Efficiency:

- server – 98.81%
- environment – 98.77%

Real distributed computing practice (2)

Virtual screening, i.e. searching for inhibitors for cancer proteins (together with “Molecular Technologies” company)

Source:

- 41 tasks, 318 portions in each
- average time of portion processing - 6 hours

Computing resources – SKIF MSU “Chebyshev”:

- simultaneous processing of several tasks
- 160 cores (40 nodes) per each task
- working together with Cleo batch system

Time:

- total – 7.5 days
- CPU – 10.6 years

Contacts

- http://X-Com.parallel.ru (in Russian)
  - latest versions:
    - http://x-com.parallel.ru/download/xcom2.tar.gz
    - http://x-com.parallel.ru/download/xcom2.zip
- x-com@parallel.ru