

# **DLB – A Dynamic Load Balancing Tool for Grid Computing**

*R.U. Payli, E. Yilmaz, H.U. Akay, A. Ecer, S. Chien*

*Computational Fluid Dynamics Laboratory  
Purdue School of Engineering and Technology  
Indiana University-Purdue University Indianapolis  
Indianapolis, Indiana 46202 USA*

*<http://www.engr.iupui.edu/cfdlab>*

# Outline

- ? **Dynamic Load Balancing**
- ? **Software Architecture of DLB**
- ? **System Requirements of DLB**
- ? **DLB Input Files**
- ? **How to use DLB**
- ? **Results**

# Dynamic Load Balancing (DLB)

- ? DLB is used to provide application level load balancing for individual parallel jobs.
- ? It ensures that all loads on the environment are distributed in such a way that overall load in the system is balanced and application programs get maximum benefit from available resources

# Software Architecture of DLB

- ? Major components of the DLB
  - System Agent
  - DLB Agent, and
  - DLB Balancer
- ? System Agent and DLB Agent are written with Java. DLB Balancer written with C++.

# System Agent

- ? System Agent acts as a server. A copy of System Agent has to run on all machines where users want to run their applications.
- ? System Agent provides system load information about the machine which is running.
- ? Also provides communication timing between two hosts.

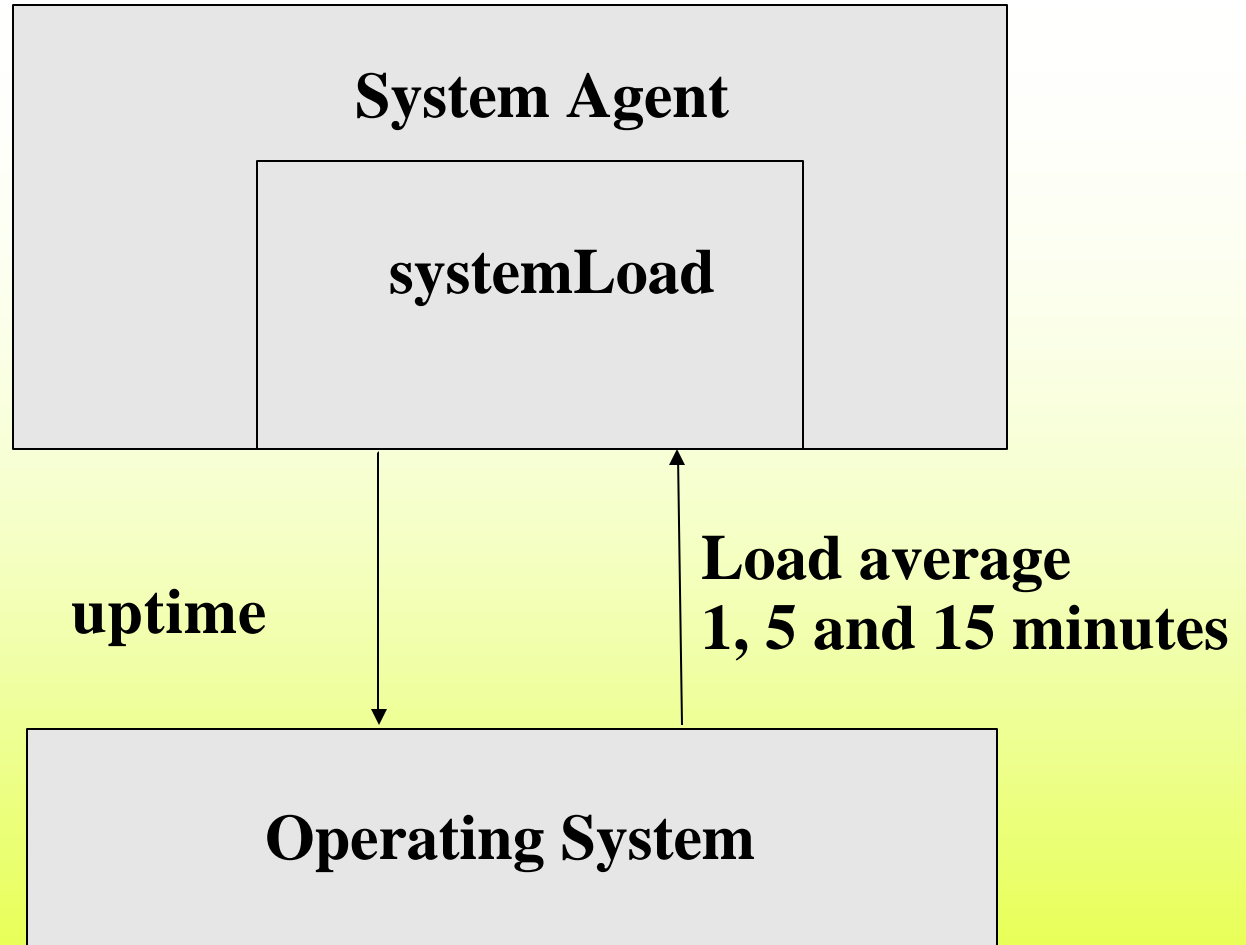
# System Agent

- ? It has systemLoad and roundTripTime methods to provide load and communication timing information.
- ? systemLoad method issues Linux/Unix “uptime” command and gets the last 1minute, 5minutes and 15 minutes system load average which given by the “uptime” command. And calculates the average of this three values to determine the system load.

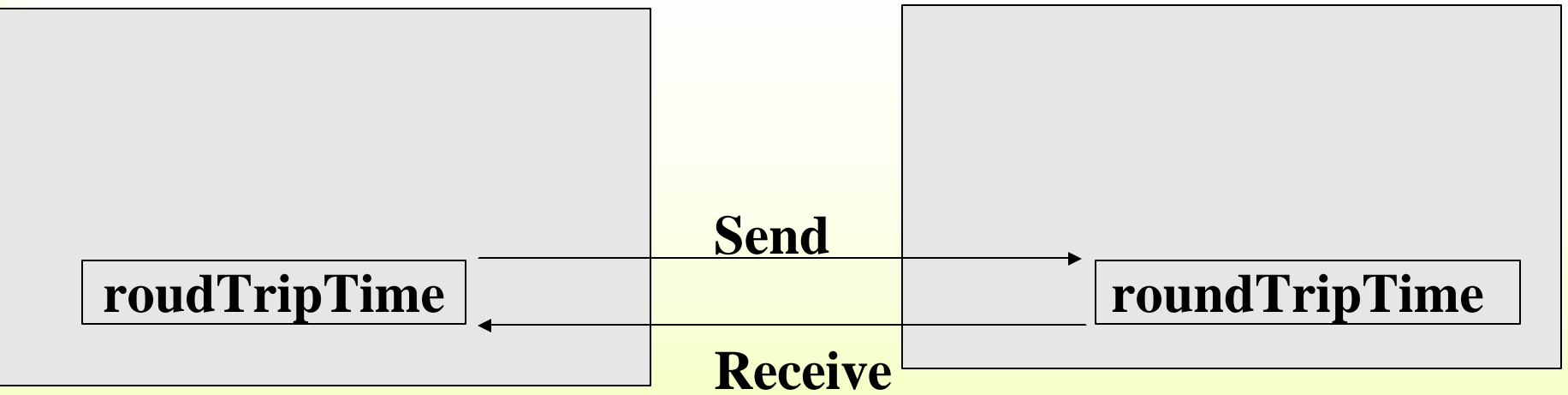
# System Agent

- ? roundTripTime method calculates the round trip time of a 1024 bytes message sending and receiving between source and destination hosts.

# System Agent



# System Agent



# DLB Agent

- ? DLB Agent prepares system load information file and communication time file for the DLB Balancer module.
- ? DLB Agent has 3 methods to prepare these two files. These methods are:
  - measureCommunicationLatency
  - getCommunicationInfo
  - getSystemLoadInfo

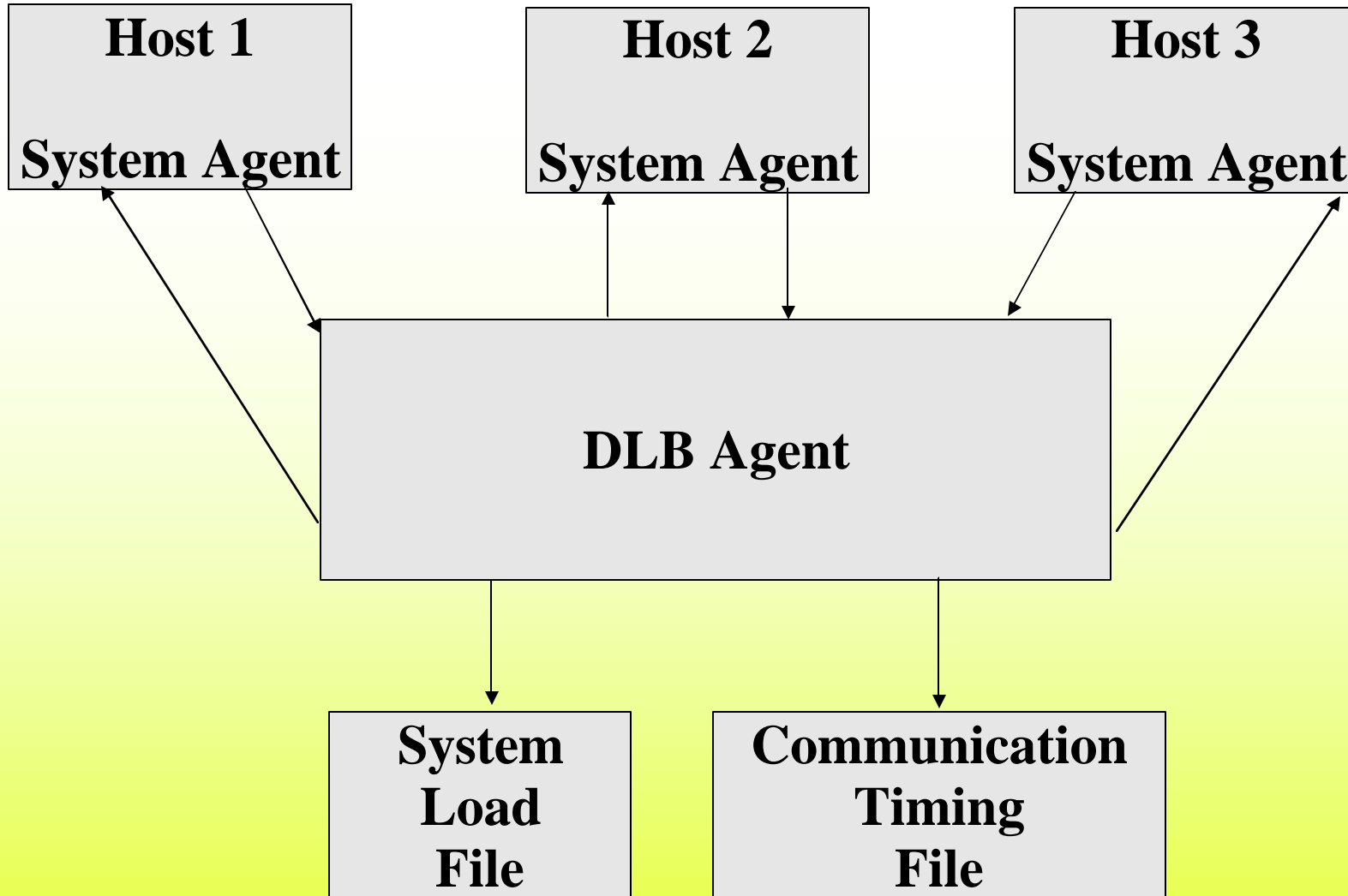
# DLB Agent

- ? measureCommunicationLatency method contacts System Agent on the each host which are in the host information file of the user and calculates the communication time between each pair of hosts using the System Agent's roundTripTime method.
- ? getCommunicationInfo method retrieves the communication time info for each pair of hosts which is calculated by the measureCommunicationLatency method and generates the communication info file.

# DLB Agent

- ? getSystemLoadInfo methods contacts all the System Agents which are in the user's host lists and gets the system load information which is calculated by the System Agent's systemLoadInfo method and generate the system load information file.

# Architecture of DLB



# DLB Balancer

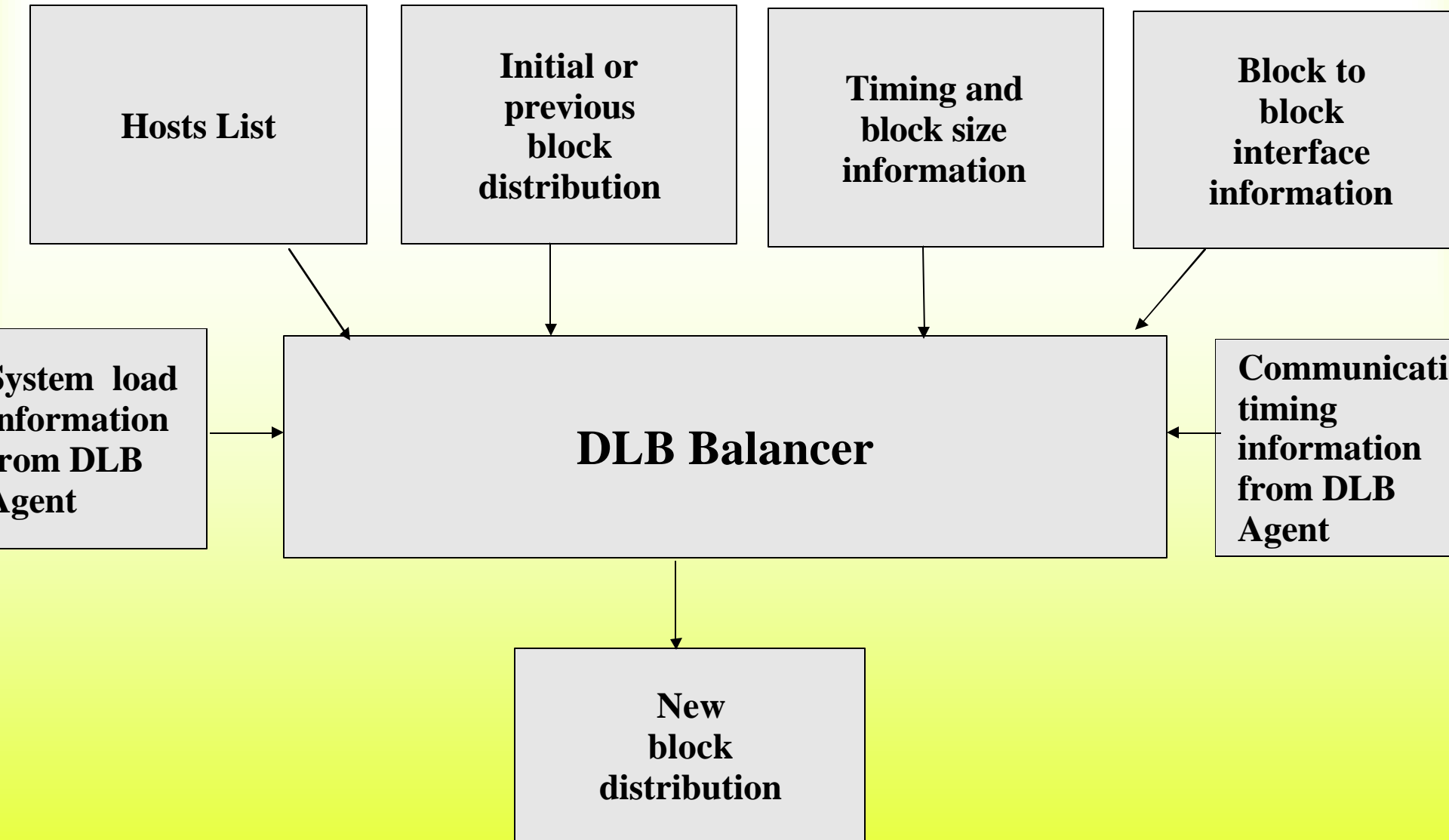
- ? DLB Balancer does the load balancing based on the load and communication timing information of the machines, the application's timing and interface information.
- ? Load Balancer moves to a process(block) from the loaded machine to the least loaded machine to reduce total elapsed time. To do this Load Balancer use Greedy Algorithm.

# Architecture of DLB

## Greedy Algorithm

- ? A Greedy Algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. This approach select the best choice at each step. Among this steps one of them is optimum.

# Architecture of DLB



# System Requirements of DLB

- ? LINUX/UNIX Operating Systems.
- ? Java 1.4 is recommended for System Agent and DLB Agent.
- ? DLB Balancer requires C++
- ? User can use any implementation of the MPI

# DLB Input Files

- ? User has to provide **four input files** for DLB. These files are:
  - Hosts file : list of available computers
  - Procdef file: block distribution on computers
  - Timing file: elapsed time for each block
  - Interface file: mesh interface size and neighbours
- ? Any parallel application can use DLB provided that those four files are generated by user.

# DLB Input Files

- ? Hosts File: Each lines of this file contains host name which user wants to use in the simulation of his/her problems. Not all processors need to be used by user but it will be available to DLB balancer in block redistribution process.
- ? Example: Full machine names are given

**mech20.cfd.iupui.edu**  
**mech21.cfd.iupui.edu**  
**aegean01.cfd.iupui.edu**  
**aegean02.cfd.iupui.edu**

# DLB Input Files

- ? Procdef File: This file contains the host names which is the processes(blocks) are running. Contrast to the Hosts File same host name can be repeated in this file which indicates that more than one process(block) are running in this host
- ? Example: This file shows process distribution for an MPI application. User should refer to MPI user manuals for details. Entries are machine name, number of process at each machine (0 for submitting machine), executable name with full path, and user name.

```
mech21.cfd.iupui.edu 0 /usr/local/cfd01/eyilmaz2/Pacer3D/Dlb/pacer3d_aix eyilmaz  
mech20.cfd.iupui.edu 1 /usr/local/cfd01/eyilmaz2/Pacer3D/Dlb/pacer3d_aix eyilmaz  
aegean01.cfd.iupui.edu 1 /cfd01/eyilmaz2/Pacer3D/Dlb/pacer3d_linux eyilmaz  
aegean02.cfd.iupui.edu 1 /cfd01/eyilmaz2/Pacer3D/Dlb/pacer3d_linux eyilmaz
```

# DLB Input Files

- ? Timing File: Each line of this file contains process(block) number, grid size of the block and total elapsed time (as a seconds) to solve this block. User (or application programmer) should provide necessary implementation to extract relative elapsed timing for each block. Prior to the first run of DLB, Timing File is generated by application program.

- ? Example:

```
1 1125 87
2 1240 94
3 1057 79
4 1189 91
```

- ? For subsequent DLB cycles this file is to be regenerated by the application program.

# DLB Input Files

- ? Interface file: Each line of this file contains source block number, destination block number, and the size of the message as a byte.

**Example:** Lets say between block 0 and 1 you have 500 grid points on the interface and you have 5 single precision (4 bytes) solution variables. Your message size will be 10000 bytes (500x5x4) and the Interface file looks like as follows

```
0 1 10000
0 2 11500
1 2 11060
1 3 10900
```

# How to use DLB

- ? If it is not already running, start the System Agent on all the machines which you would like to use it.
- ? Run your parallel application and get the Timing File.
- ? Run DLB Agent and DLB Balancer to get the new process (block) distribution.

# How to get Timing:

This is simplified coding of the main program for the application used here. MPI is the parallelization library used for this program, therefore timing function provided by MPI is used to get elapsed time for block. Note that communication time is not included in block time, therefore it is subtracted at the end from the total time.

## Colors:

**Black:** application coding

**Green:** MPI related coding

**Blue:** timing info implementation

## Program Pacer3D

```
.....
! initialize parallel environment
  call mpi_initialize()
  myid = mpi_comm_rank(...,myid,...)
.....
! read grids
  call gridRead()
! get start time for block
  eltime1blk = mpi_wtime()

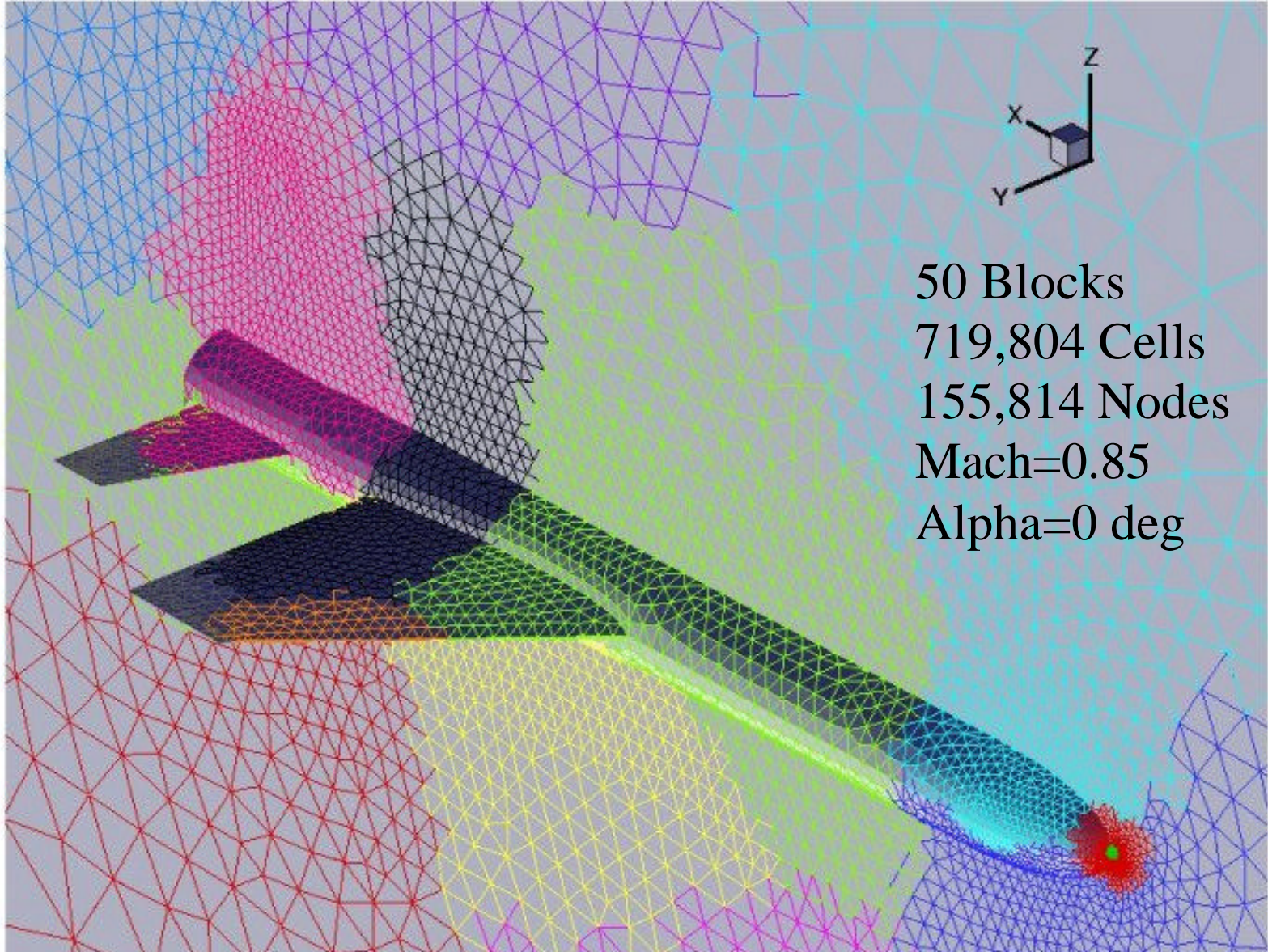
! time stepping
  Do iTime=1, nTime
    call timestep()
    call boundary()
    call fluxes()
! get communication elapsed time
    eltime1int = mpi_wtime()
    call interface()
    eltime2int = mpi_wtime()
    eltimeint = eltimeint + (eltime2int-eltime1int)
    call residuals()
  EndDo

! get final time and total elapsed time for block
  eltime2blk = mpi_wtime()
  eltimeblk = eltime2blk-eltime2blk
  eltimeblk = eltimeblk - eltimeint
! write time information into timing file
  write(2,*) myid, nelelem, eltimeblk

.....
  call mpi_finalize()
.....
End
```

# Test Case

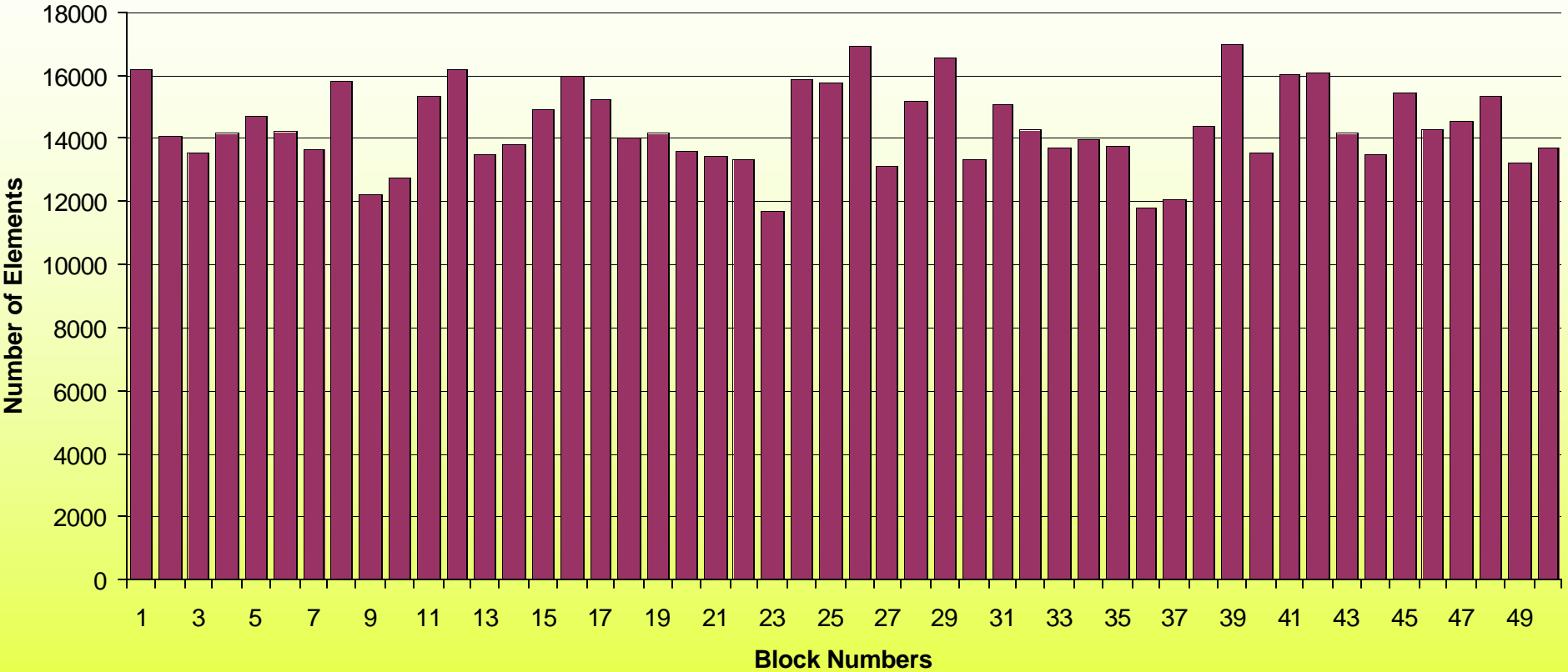
## 3D Euler Solver, Generic Missile Geometry



# Test Case

## Generic Missile Geometry: Parallel Block Size

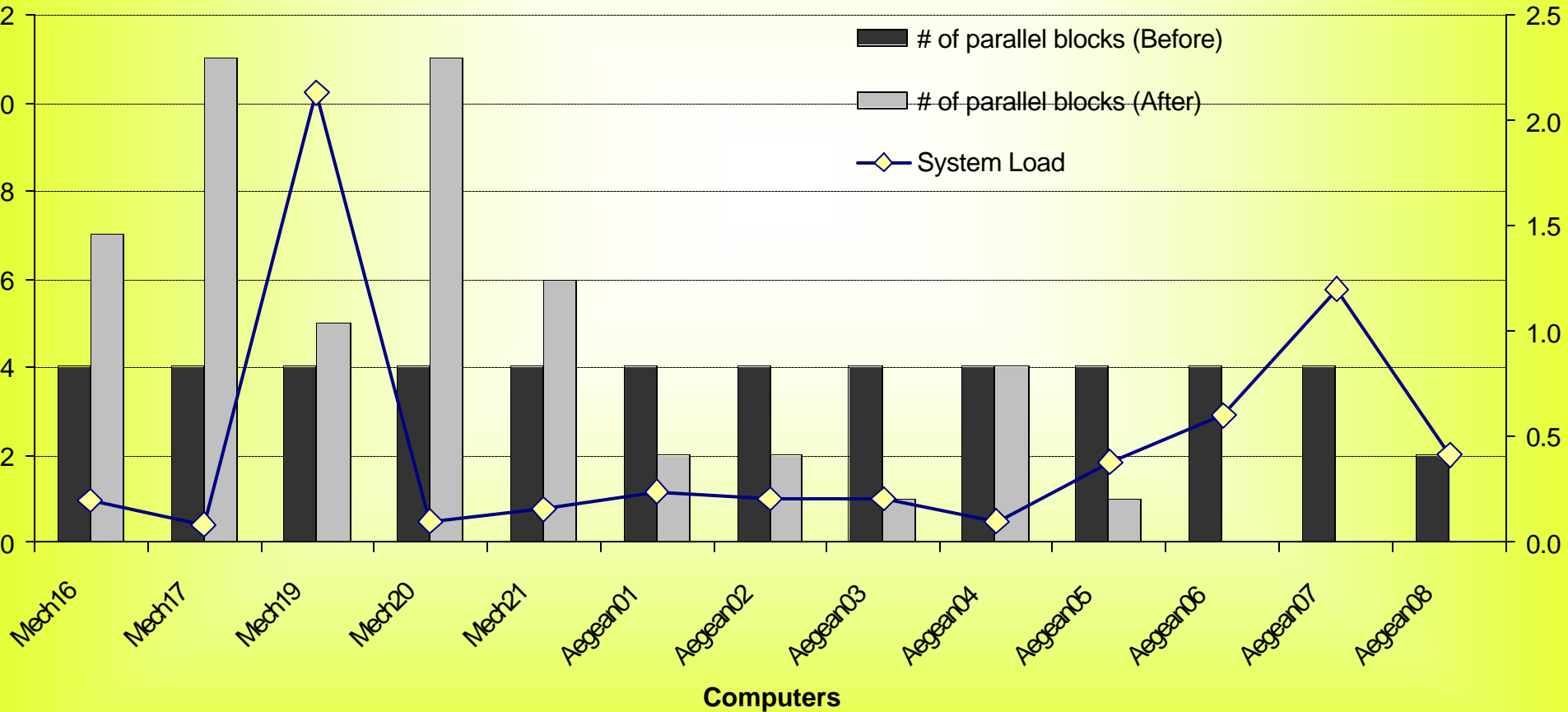
Generic Missile Geometry Block Size



# DLB Results

## Block Distribution Before and After DLB

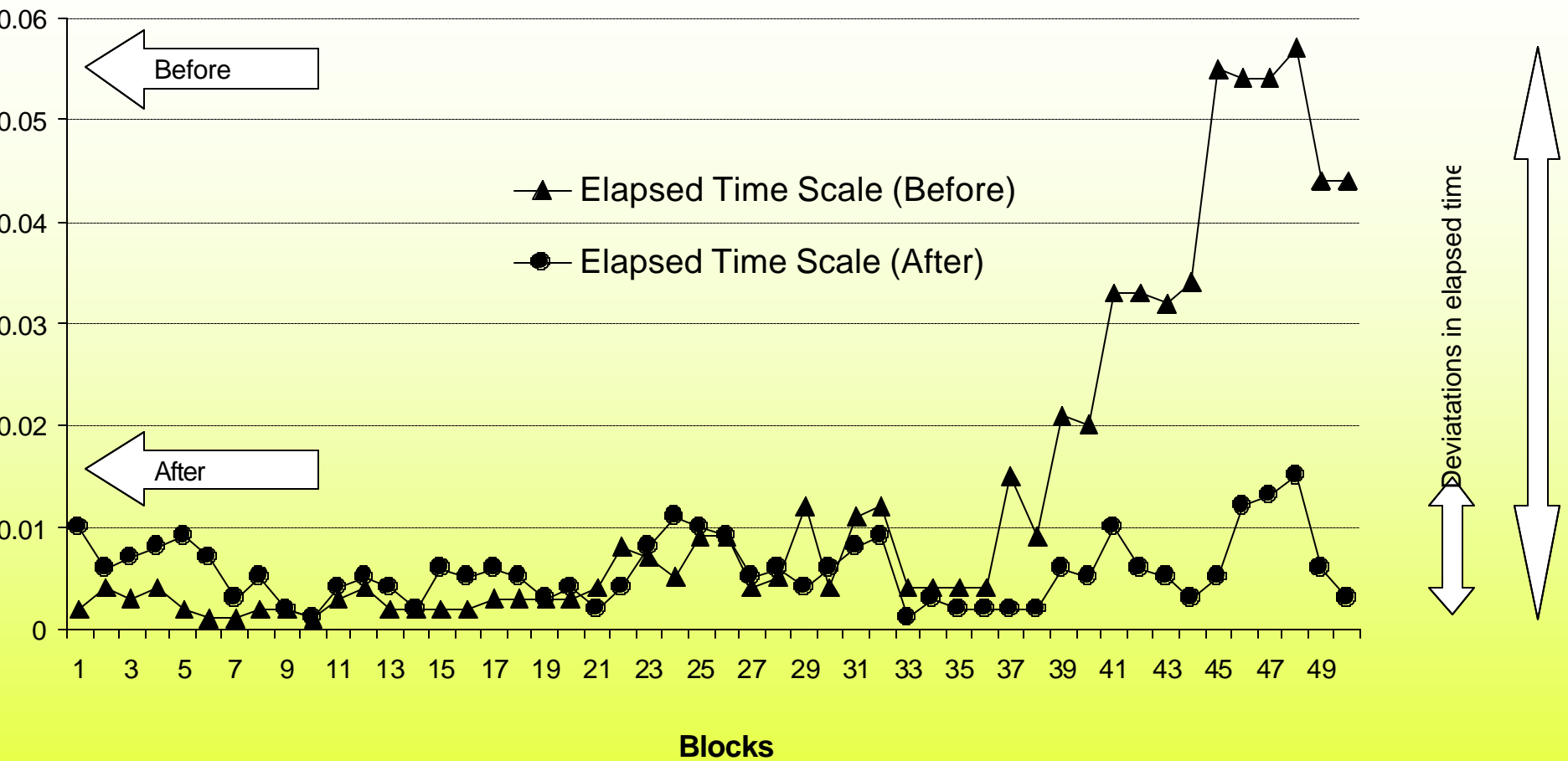
Block Distribution Among the Computers Before and After DLB



# DLB Results

## Elapsed Time Before and After DLB

Elapsed Time For Blocks Before and After DLB



# Conclusion

- ? DLB components are simplified for application programmers and users.
- ? It can be used by parallel applications by providing necessary input files.
- ? Test case shows DLB makes better distribution right after first run.

# Future Work

- ? Add interface parts of DLB to this simplified form for automatic run.
- ? Provide new test case, attached to DLB package, this test case will be same as PCF test case provided before but grid partitioning will be made in all Cartesian directions.

# Contacts

To obtain DLB package and further details please contact to following persons:

Purdue School of Engineering and Technology,  
IUPUI, Indianapolis-IN, USA

- Akin Ecer: [aecer@iupui.edu](mailto:aecer@iupui.edu) , or
- Resat U. Payli: [rpayli@iupui.edu](mailto:rpayli@iupui.edu)