Introduction to MPP version of LS-DYNA®

Part I

Jason Wang
08/09/2012

Livermore Software Technology Corporation

Disclaimer

The material presented in this text is intended for illustrative and educational purposes only. It is not intended to be exhaustive or to apply to any particular engineering design or problem. Livermore Software Technology Corporation assumes no liability or responsibility whatsoever to any person or company for any direct or indirect damages resulting from the use of any information contained herein.

Contents

- Introduction LS-DYNA SMP and MPP
- Scalabillity of MPP-DYNA
- Special Decomposition
- Restart and Pre-decomposition
- General Guidelines and Debugging
- Recent Development

Introduction LS-DYNA SMP and MPP

Introduction

- Development History
- What drives the MPP development?
- Implementation of SMP and MPP
- Implementation in Production
- Numerical Variation
- Performance Comparison between SMP and MPP

Development History

- Public domain DYNA3D, Dr. John O. Hallquist/Lawrence Livermore National Laboratory, 1976
  - Weapon simulations
- LSTC and LS-DYNA3D® founded by Dr. J. O. Hallquist in 1988
  - Recognized market for commercial applications
- In the 1990's …
  - LS-DYNA2D and LS-DYNA3D® combined (LS-DYNA)
  - Implicit capability (LS-NIKE3D) introduced to LS-DYNA®
  - Thermal capability (TOPAZ) introduced to LS-DYNA®
  - Introduced MPP capability
  - Eulerian/ALE element formulations and Euler/Lagrange coupling introduced
  - LS-POST, LS-OPT® introduced

Development History

- Since 2000…
  - Expanded MPP capability
  - Meshless methods introduced (SPH, DEM, EFG, etc.)
  - Integrated Multiphysics Solvers (CFD, EM, etc.)
  - LS-POST expanded to include preprocessing (LS-PrePost®)
- Worldwide distribution: US, UK, Nordic countries, France, Germany, Italy, Netherlands, Japan, Korea, China, Taiwan, India, Brazil; also through ANSYS and MSC.
- 60+ full-time employees + numerous consultants
- Products:
  - LS-DYNA®
  - LS-PrePost®
  - LS-OPT®
  - FE Models: Dummies, barriers, head forms
  - USA (Underwater Shock Analysis)

Development History

More Applications Fields

- Automotive
  - **Crash and safety**
  - Durability
  - NVH
- Aerospace
  - Bird strike
  - Containment
  - **Crash**
- Manufacturing
  - **Stamping**
  - Forging
- Structural
  - Earthquake safety
  - Concrete structures
- Electronics
  - Drop analysis
  - Package design
  - Thermal
- Defense
  - Weapon design
  - Blast response
  - Penetration
  - Underwater shock analysis
- Also, applications in biomedical, sports, consumer products, etc.

Development History

Different Physics

- Combine the multi-physics capabilities
  - Explicit/Implicit solver
  - ALE, SPH, EFG
  - Heat Transfer
  - Airbag particle method
  - Discrete Element Method
  - Acoustics (USA)
  - Interfaces for users, i.e., elements, materials, loads
  - Electromagnetic (version 981)
  - Incompressible fluids (version 981)
  - CESE compressible fluid solver (version 981)

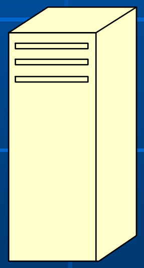
- into one scalable code for solving highly nonlinear transient problems to enable the solution of coupled multi-physics and multi-stage problems.

  → MPP

Development History

- SMP (Shared Memory Parallel)
  - Start and base from serial code
  - Using OpenMP directives to split the tasks
  - Only run on SMP (single image) computers
  - Scalable up to ~8 CPUs (Depends on model – see next slide)

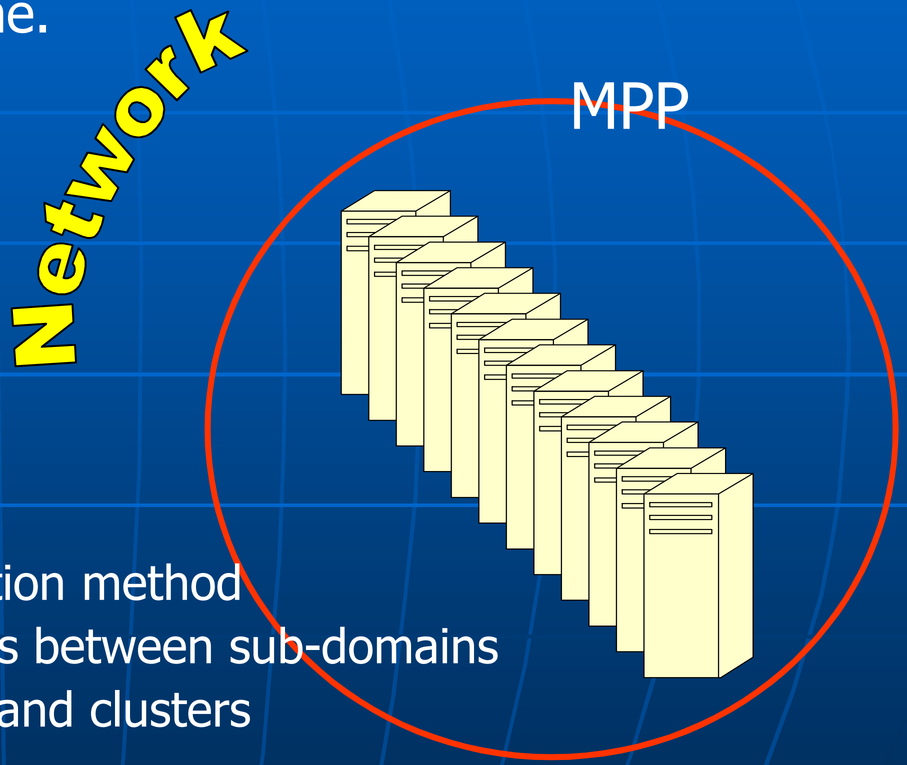
SMP



SMP can run multiple CPU's but
they are placed in the same computer

Development History

MPP is a special version of LS-DYNA®, that is developed to run on a number of computers connected in a network. For large models this it is necessary to have large computer resources to finish a simulation in an acceptable time.



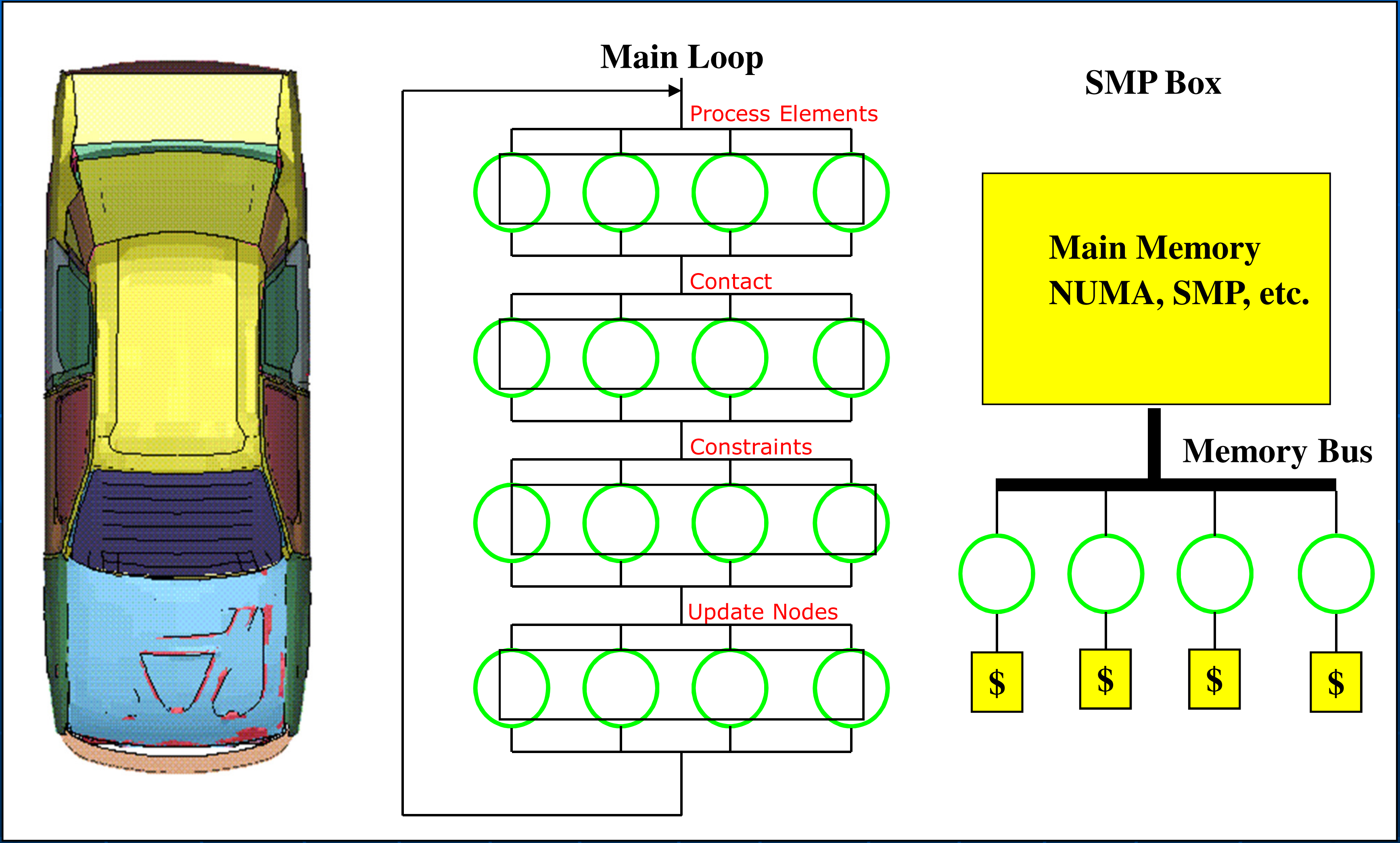
- MPP (Message Passing Parallel)
  - Using the domain decomposition method
  - Using MPI for communications between sub-domains
  - Work on both SMP machines and clusters
  - Scalable >> 8 CPUS
  - Dramatically reduced elapsed time and the simulation cost

Development History

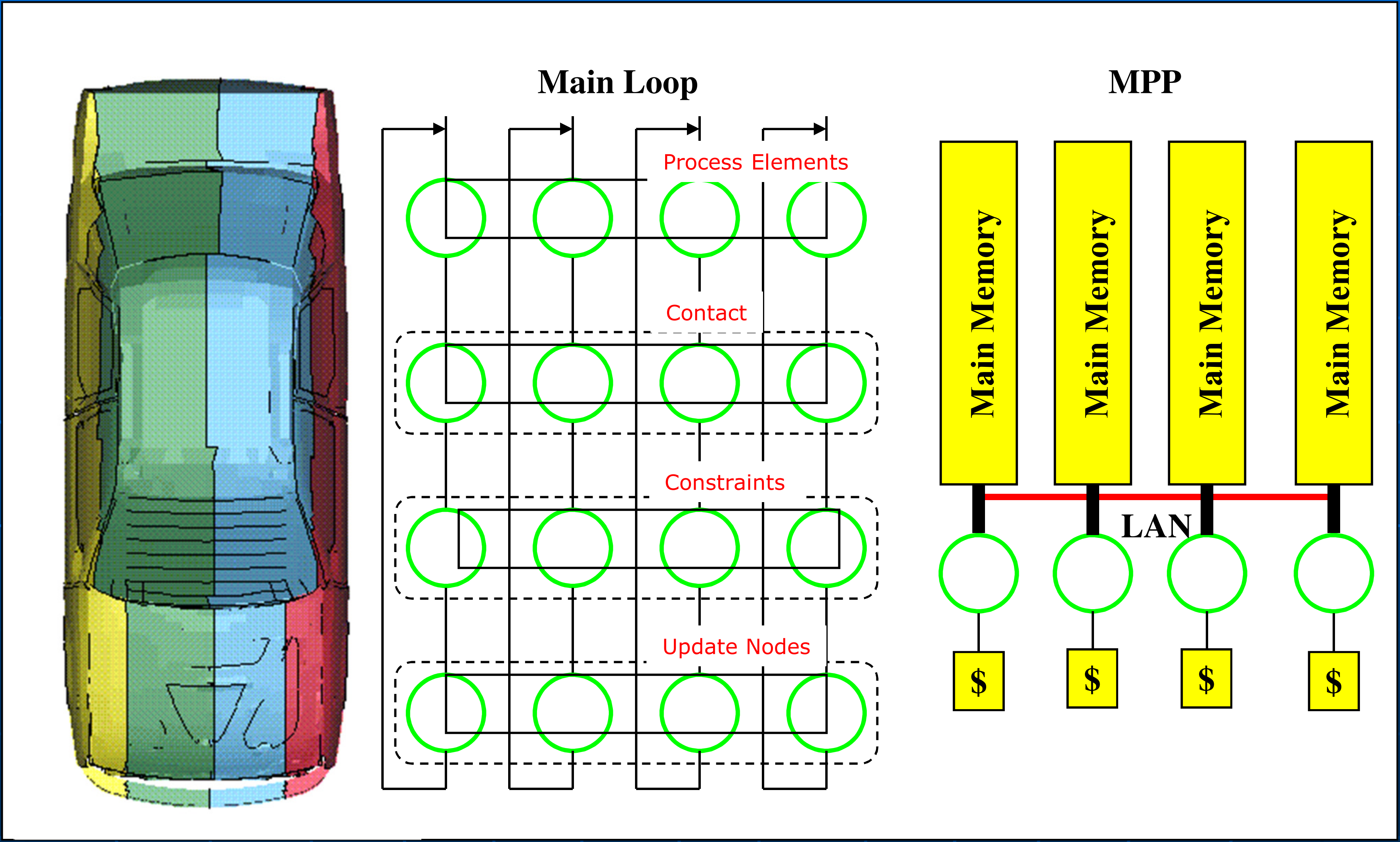
Many of the features were implemented as customers required it. This means that features were not implemented in option blocks.

- MPP-DYNA was initiated in 1993 (version 930)
- Nearly fully supported contact algorithms (1996)
- P-file, composition and analyze in one run (1996)
- CONSTRAINED_options (1996)
- Limited ALE capabilities (1998)
- SPH (2002)
- EFG (971)
- Thermal (971)
- Constantly development, recently some feature first in MPP, before they appears in MPP!

Implementation of SMP Parallelism



Implementation of MPP Parallelism



Implementation of SMP and MPP

- SMP
  - Long history of production use
  - Stability
  - Rich features and many advanced new features
  - Easier for most of developers

- MPP
  - New algorithms
  - Parallelism requires new algorithms
  - Some features unsupported
  - Better speedup

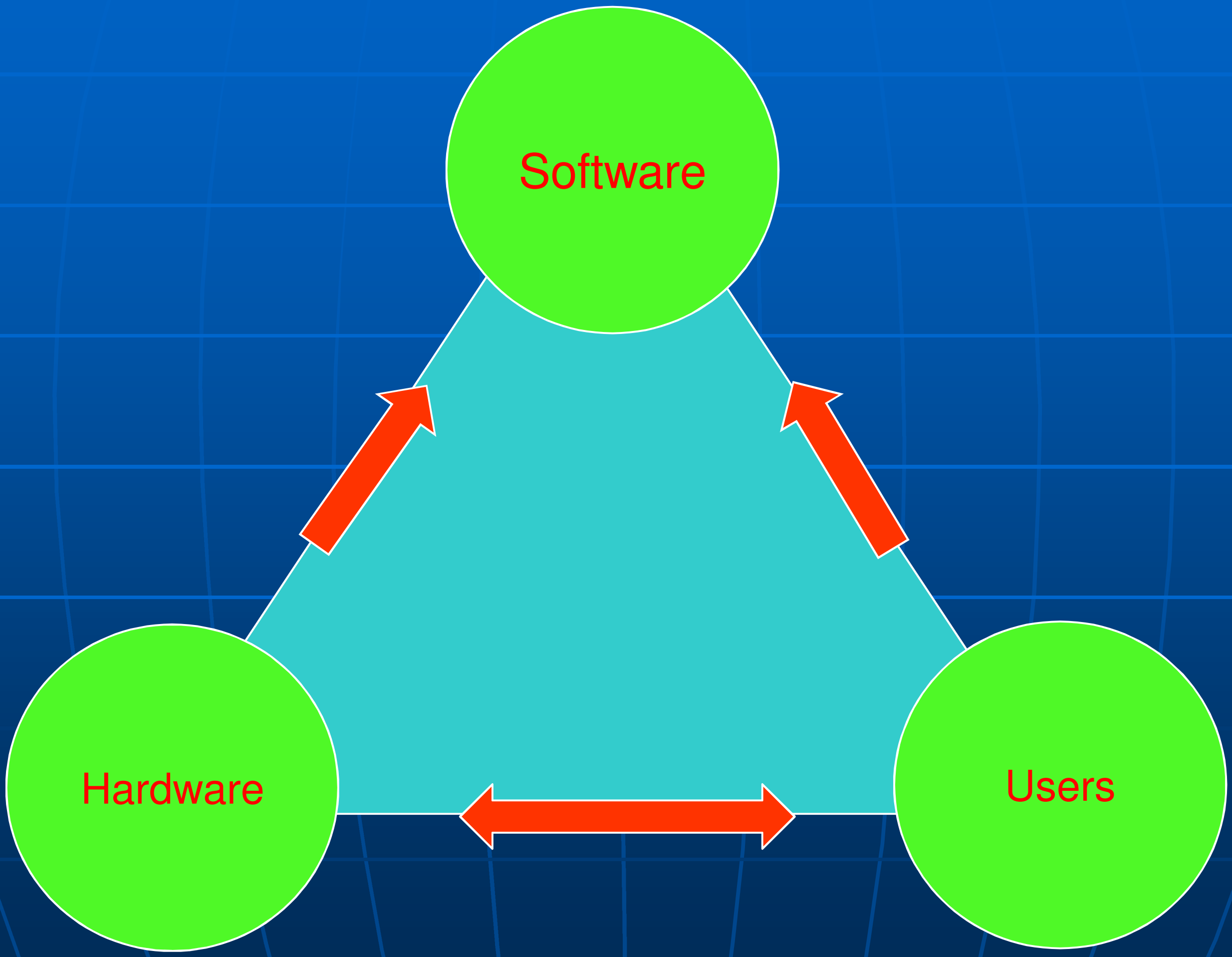
Implementation of SMP and MPP

Some of the Different Routines

- *AIRBAG_
- *ALE_
- *BOUNDARY_
- *COMPONENT_
- *CONTACT (major – see “Contact” Section)
- *CONSTRAINED_
- *DAMPING_
- *DATABASE_
- ....

But *ELEMENT_ and *MAT_ are the same !!

What Drives the MPP Development?



What Drives the MPP Development?

- MPP development is mainly driven by the automotive and the aerospace industry
  - Crash test – impact of vehicle
  - Airbag deployment – control volume, ALE, and CPM
  - Manufacturing of parts – primarily Sheet Metal Forming
  - Hydroplaning – ALE
  - Bird impact – ALE / SPH
- Military applications
  - Explosions – ALE, SPH, DES
  - Penetration problems – Fine mesh

What Drives the MPP Development?

Automotive – Better Prediction

- Smaller element size
- More expensive element formulation
- Non-local failure
- Complicated spotweld capabilities (cluster of solids)
- More sophisticated material models
- Fine mesh barriers and dummies
- Crash models with stamped parts

➡ Longer simulation time

What Drives the MPP Development?

Automotive & Military – More sophisticated problem

- Multi-physics: ALE + FSI - airbag, fuel tank
- Multi-physics: EM + metal forming
- Bio-dummies
- Explicit/Implicit analysis

→ Much longer simulation time

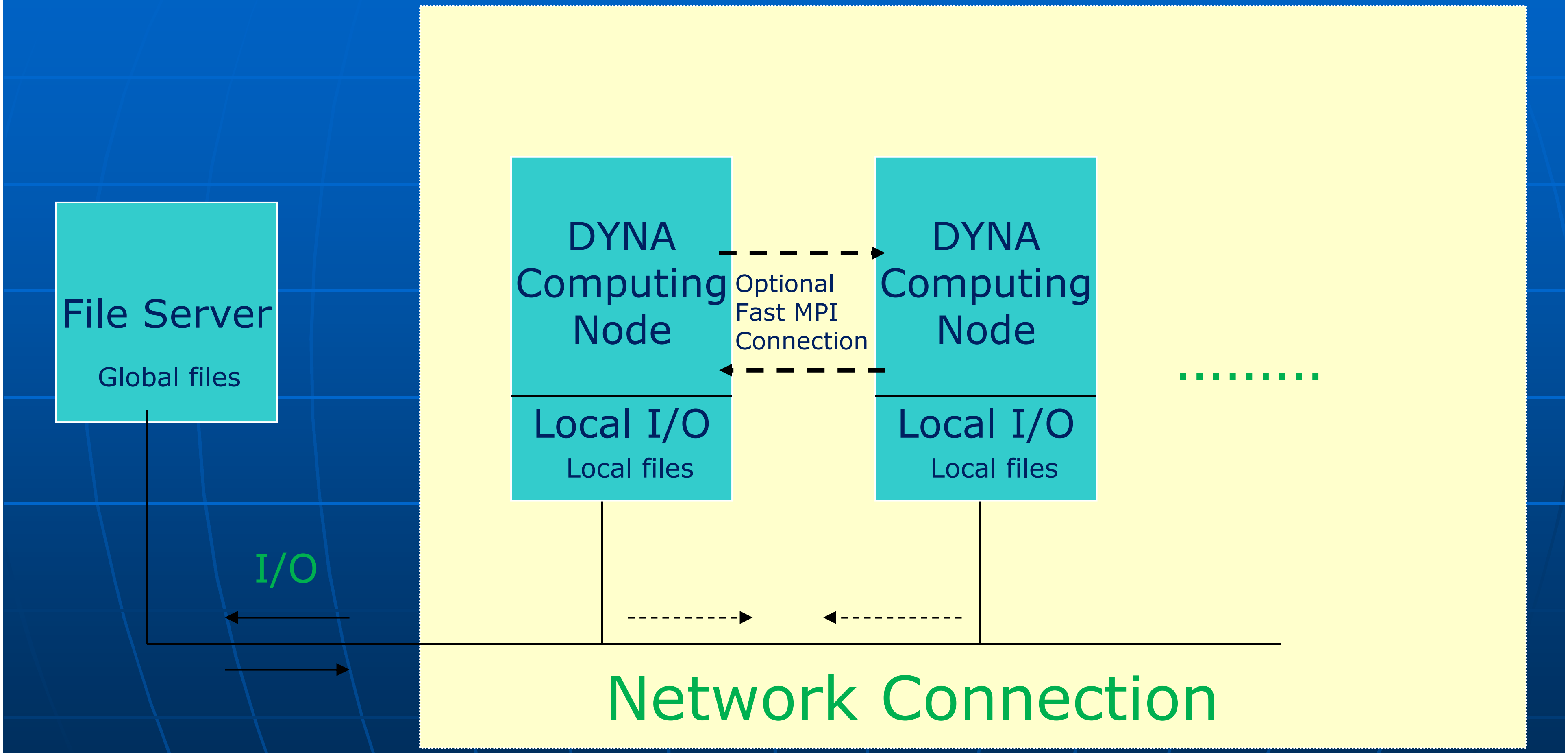
What Drives the MPP Development?

Mass Production - Cost reduction

- Produce more durable end products
- Save raw material in production line
  - few grams per product but save millions dollars in production
- Product cycle reduced from 1 year to 3 months
- Turn around time in few hours

What Drives the MPP Development?

Computing environment



File Server

Global files

DYNA Computing Node

Local I/O

Local files

Optional Fast MPI Connection

DYNA Computing Node

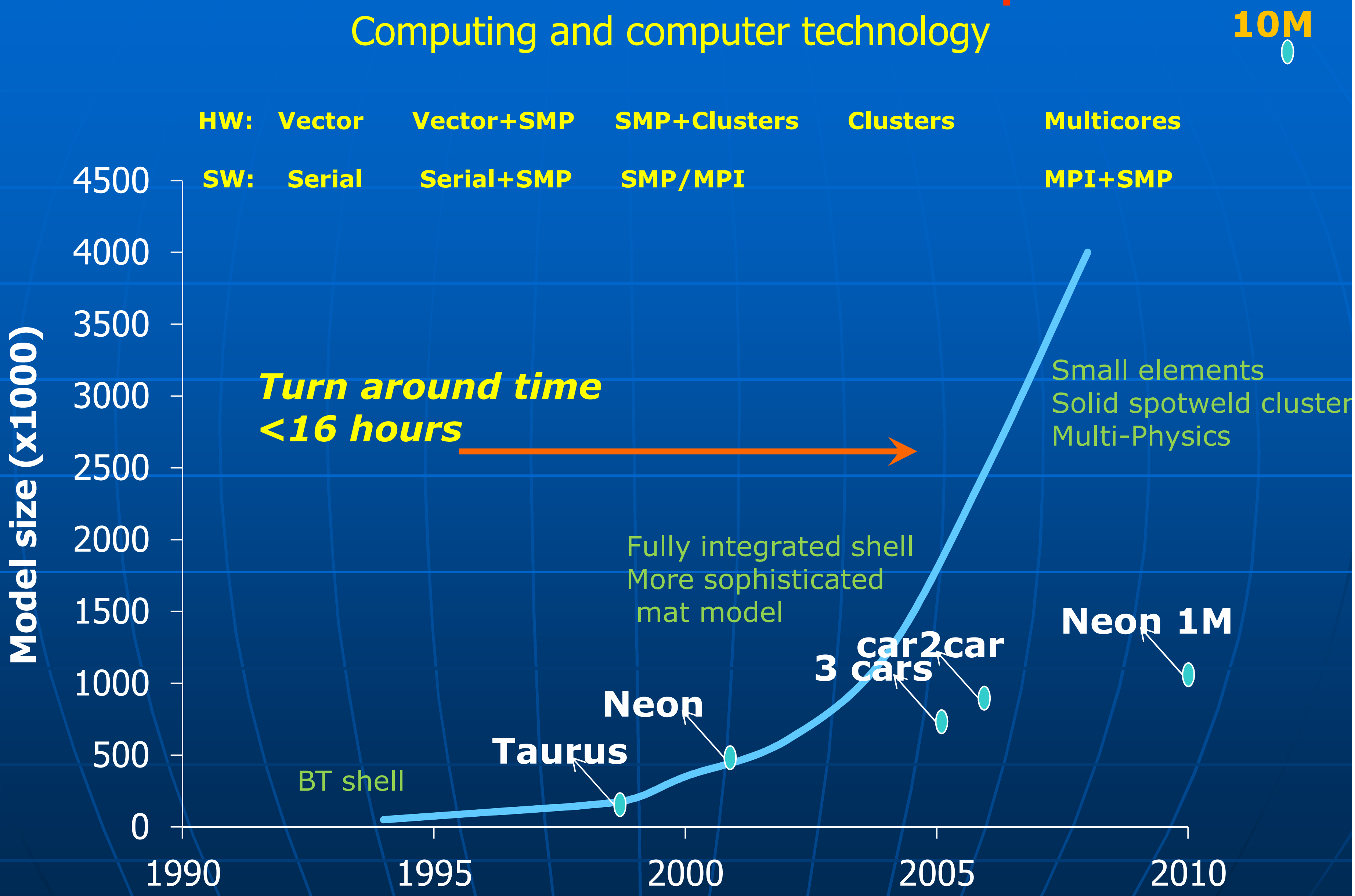
Local I/O

Local files

Network Connection

What Drives the MPP Development?

Computing and computer technology



Implementation in Production

Basic customer requirements

- Repeatability: Same decomposition = same answer
- Consistency between SMP and MPP
- Serial/SMP input = MPP input for zero conversion effort
- Decomposition+Solution in single run
- Single source for MPP and SMP for easier tracking bugs
- Supports all features/options in production models

Implementation in Production

- MPP project starts from 1993

- Chrysler 1998
  - Phase I (Q3/98) – 30 6-month old models
    - Check for missing features
    - SMP/MPP performance, results comparison
    - Open 2 12-processor queue
  - Phase II (Q1/99) – 20 production models
    - SMP/MPP performance, results comparison
    - Open 8 12-processor queues
  - Phase III (Q2/99) - 5 models for QA
    - SMP/MPP performance, results comparison
    - Madymo coupling
    - Open 16 12-processor queues + Open several 24-processor queues for high priority jobs

Fully production in 1999 and most jobs finished overnight

Implementation in Production

- Volvo (Q2/99)
  - Metal forming 1,000,000 model – 13.5 hours
- DiamlerChrysler early 2000
- GM, Ford in production 2001
- ***Many suppliers start to install clusters***
- Japan S and H companies
- Japan T company 2002
- P & G 2004
- Ohio H Company 2005…….
- …..
- ……

Implementation in Production

Impact of Computing Environment

- ~ 64 CPUs SMP/Vector DYNA Nodes at 1996 → 800 CPUs clusters and growing
- >$100/minute at 1996 → less $1/minute
- 3 days/job (100K elements) → overnight turn around time (1 million elements+more)
- 2009: 3 million elements – overnight!

LSTC Livermore Software Technology Corp.

Numerical Variations

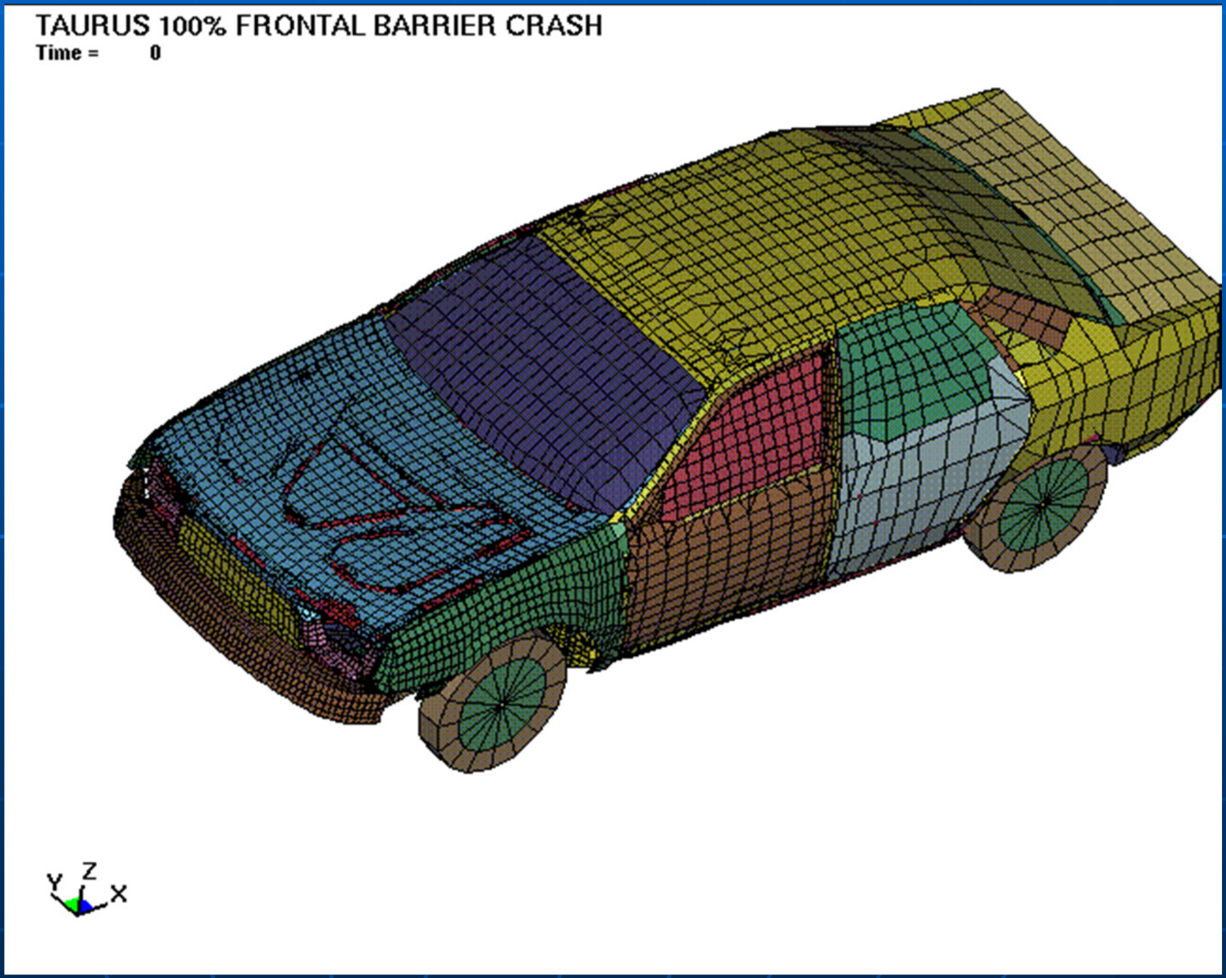
Example: Taurus to Rigid Pole

Frontal impact:

No. of materials: ~130

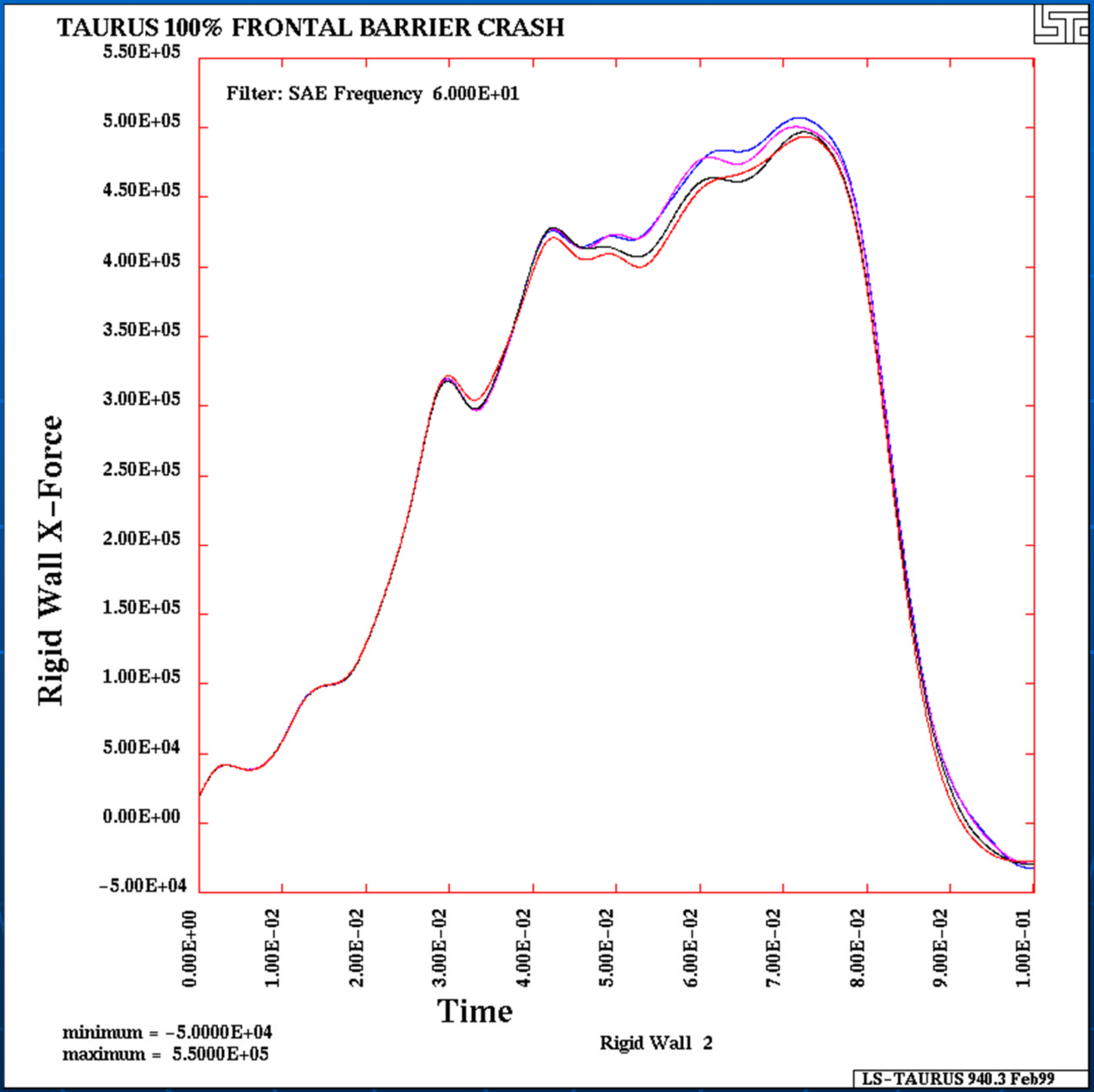
No. of shell elements: ~28,000

Simulation time: 0.10 second



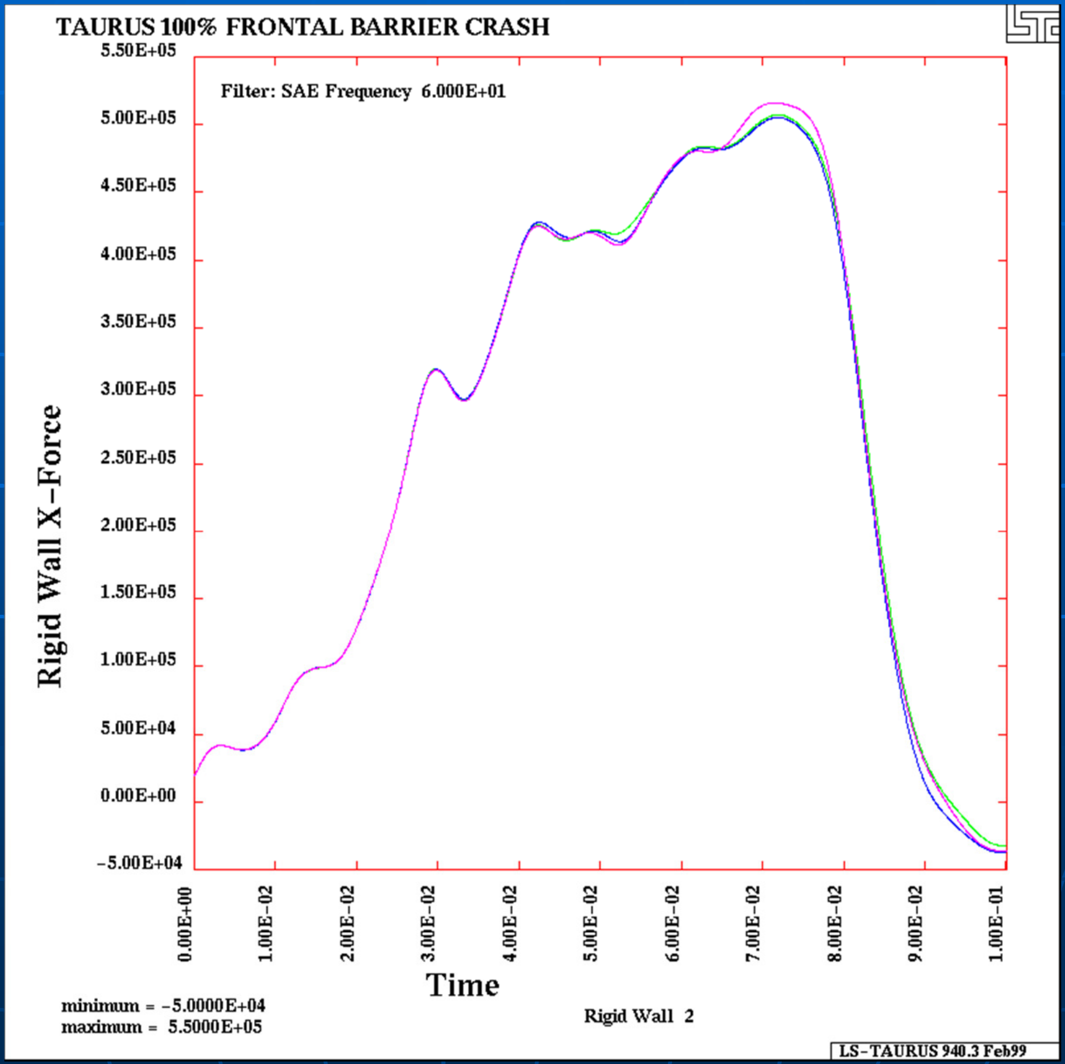
Numerical Variations

Multiple processors(MPP)/1,2,4,8 CPUs



Numerical Variations

Single Processor(SMP)/Different Platforms



Numerical Variations

- Round off error – DP may give less error
  - DP may not help, finer mesh may help
  - For OpenMP use consistency option ***ncpu=-integer***

- Changing number of processors 5% (MPP), however for a good stable model the difference is small (2009)

- Differences in MPP and SMP contact

- Look for errors in the model – different platforms handles the division by zero differently

Performance Comparison

Example: Neon Refined Model

- Frontal crash with initial speed at 31.5 miles/hour
- Model size
  - Number of nodal points: 532077
  - Number of shell elements: 535K
- Simulation length: 30 ms
- Model created by National Crash Analysis Center (NCAC) at George Washington University
  - One of the few publicly available models for vehicle crash analysis
  - Based on 1996 Plymouth Neon
  - Modified by LSTC (refined the mesh)

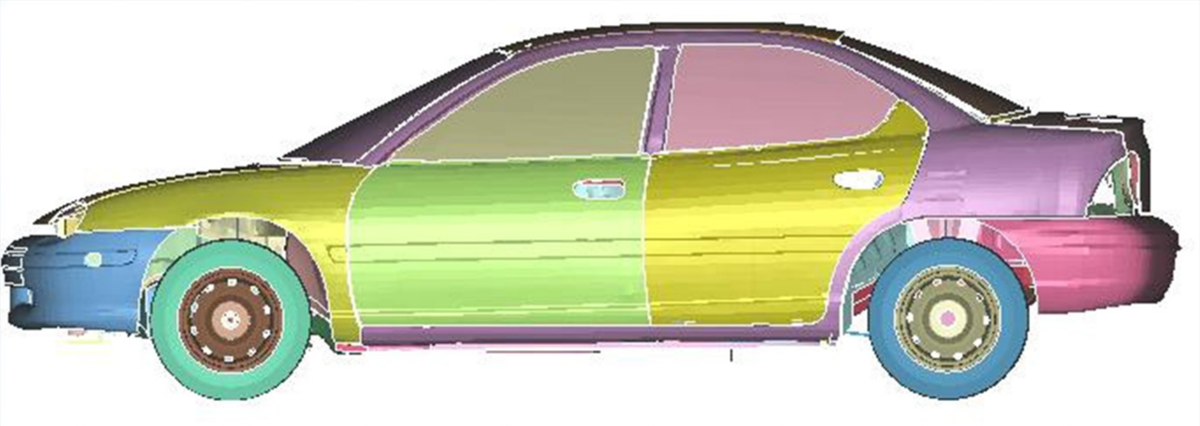
Performance Comparison

1996 Plymouth Neon

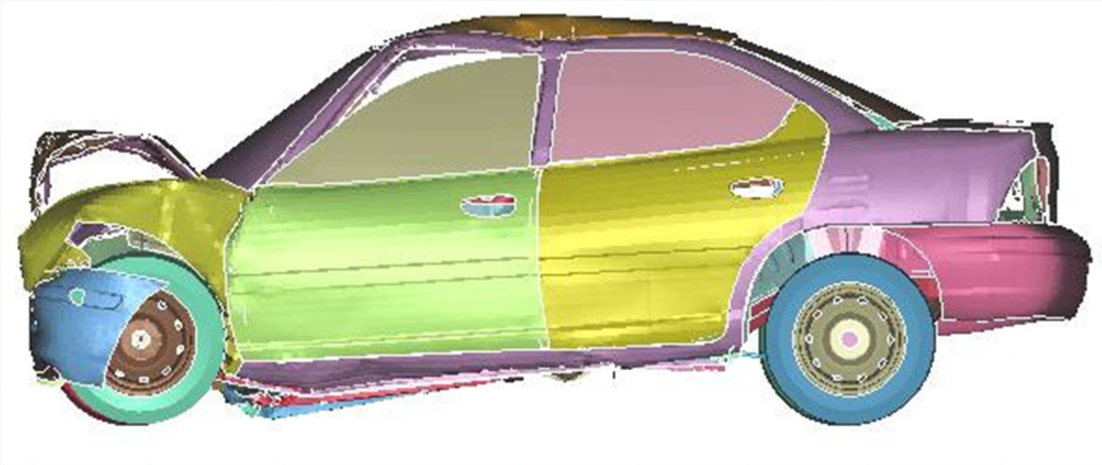
Performance Comparison

Simulation Results

Before Crash

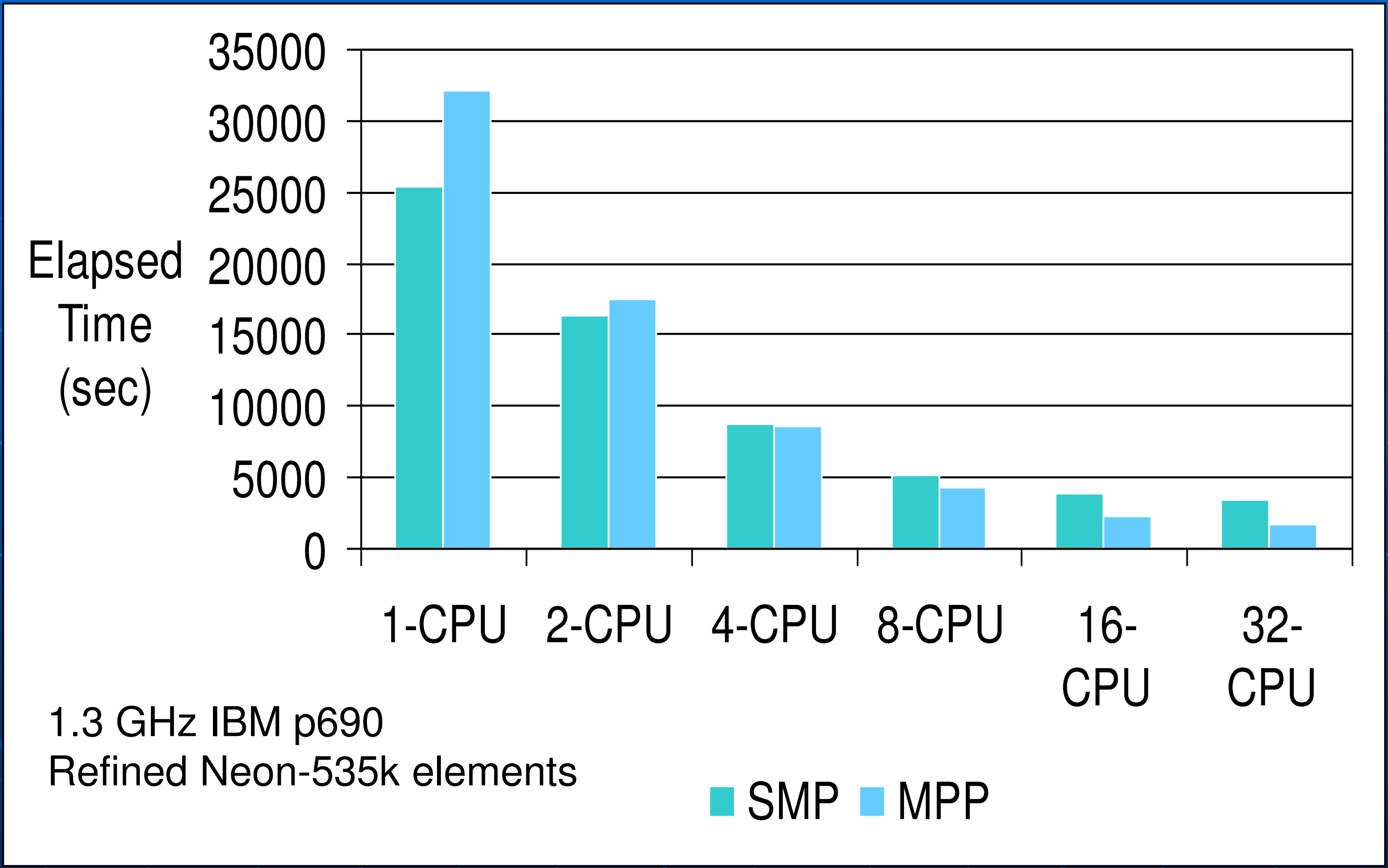


After Crash



Performance Comparison

LS-DYNA SMP and MPP

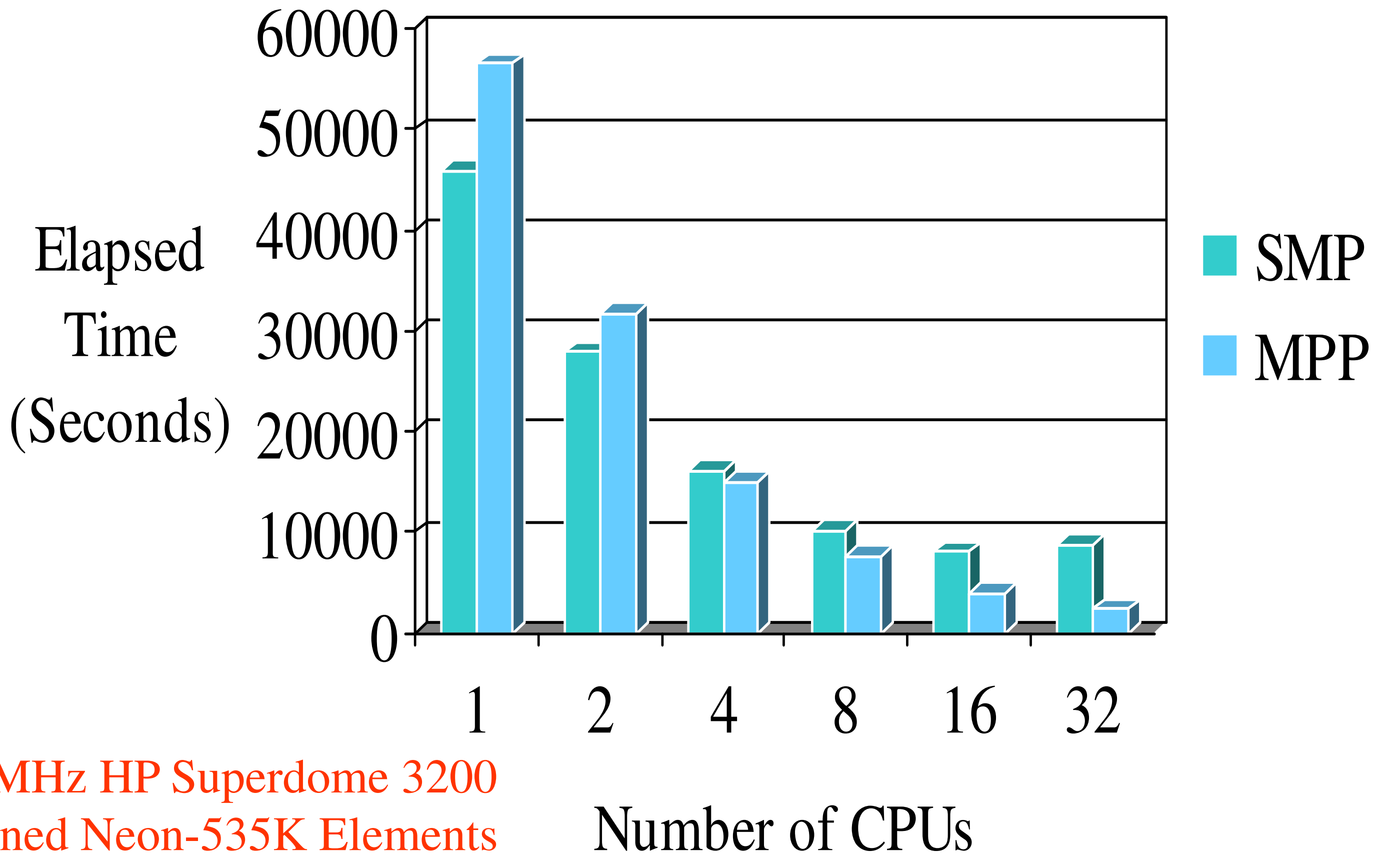


1.3 GHz IBM p690
Refined Neon-535k elements

SMP, MPP breakeven point: 2-4 processors

Performance Comparison

LS-DYNA SMP and MPP



875MHz HP Superdome 3200
Refined Neon-535K Elements

MPP-DYNA Scalability

MPP-DYNA Scalability

- Introduction
- Effects of Interconnects
- Distribution of the CPU time
- Effect of Decomposition
- Summary

LSTC Livermore Software Technology Corp.

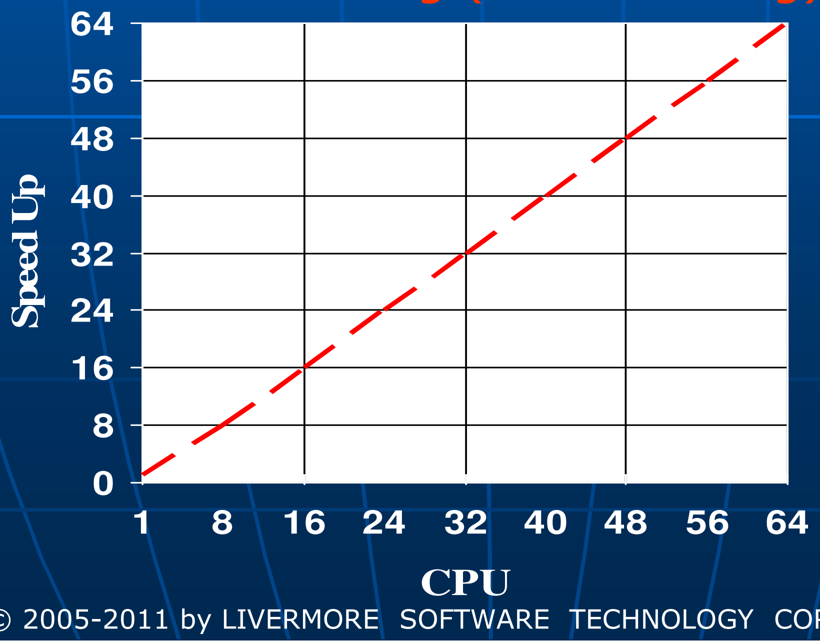
Introduction

- Scalability: *"the ability of a problem to be solved n times faster using n processors"* [Wainscott et al, 98]

- The % scalability: Can be calculated as [Galbraith et al, 2002]:
  (Elapsed time for 1 CPU / elapsed time for N CPU's) x 100/N

- Speed Up: Elapsed time for 1 CPU / Elapsed time for N CPU's

Ideal Scaling (linear scaling)



Introduction

Main factors that influence scalability/performance:

- Decomposition of the model, due to load balance (See “Decomposition” section)
- Single node computational performance
- Communication characteristics of the interconnection
  - Network: Ethernet, IB, etc
  - File system: NFS, local disks, etc
- Message Passing details
- Memory/Cache System
- Model size and problem type

Introdution

Developement of faster mashines

~493,000 elements , 370,815 cycles
LS-DYNA/MPP 960, 6/2001

CPU#	Time	Speedup
1	~21 days	1.00
4	127.03hrs	4.00
8	64.18hrs	7.92
16	32.26hrs	15.75
32	19.52hrs	26.03
64	11.05hrs	45.98
96	8.80hrs	57.74

Introduction

Why MPP-DYNA ?

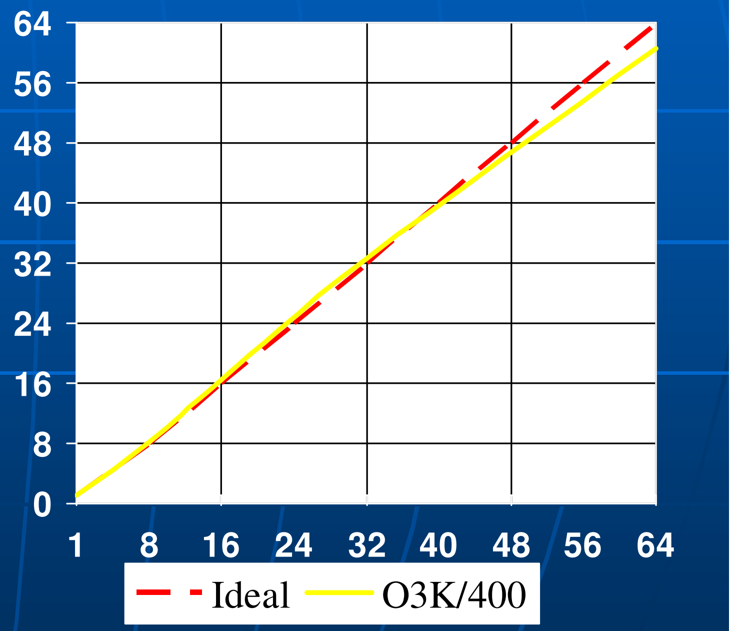
DaimlerChrysler Model w168, 429,970 elements, 100 ms simulation time. MPI Version on SGI Origin3000

Ncpu	O3K/400	Speedup
1	206 h	1.0
4	52.7 h	3.9
8	24.7 h	8.3
16	12.5 h	16.48
32	6.3 h	32.7
64	3.4 h	60.6

206 hours

↓

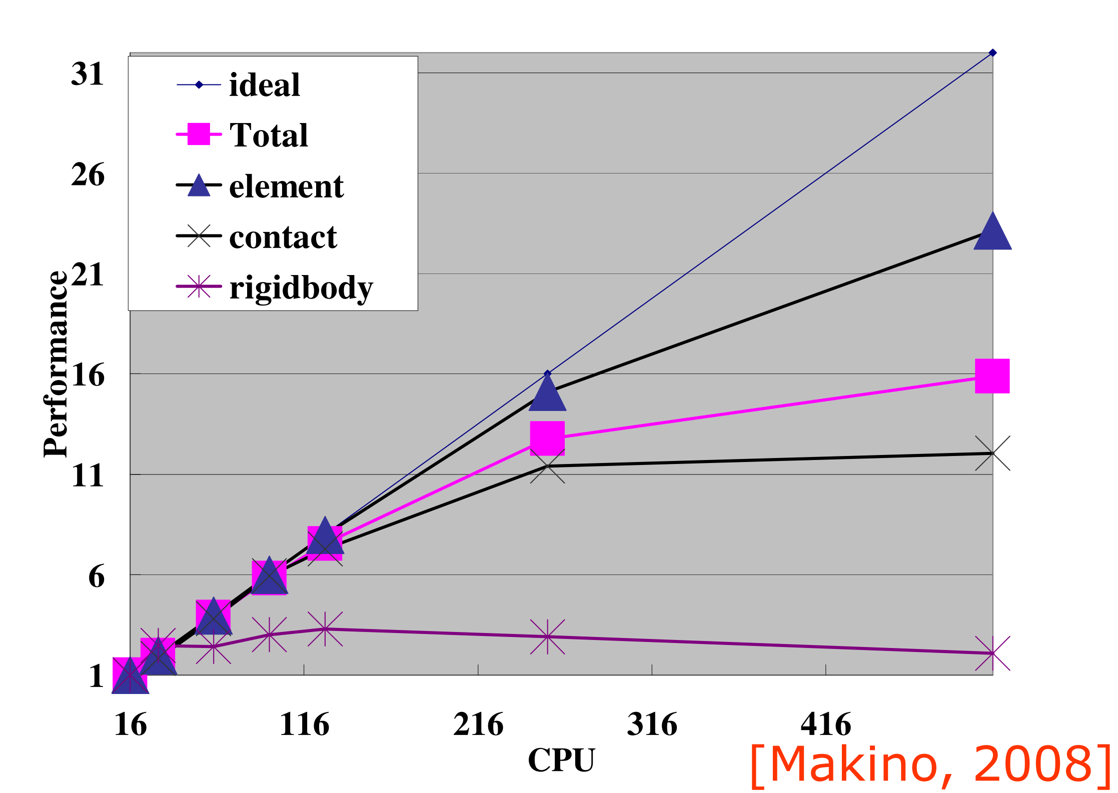
3.4 Hours



Simulation time down from 206 hours to 3.4 hours

Introduction

- The scalability depends on the numbers of CPU's. There is not an ideal scaling for a large numbers of CPU's.



[Makino, 2008]

- However, the new Hybrid version shows very promising results. Results are shown in the "Resent Development" Section.

Effects of Interconnects

- Communication is split up into:

  Explicit/Implicit Implicit

  T_elapsed = T_computation + T_communication + T_IO

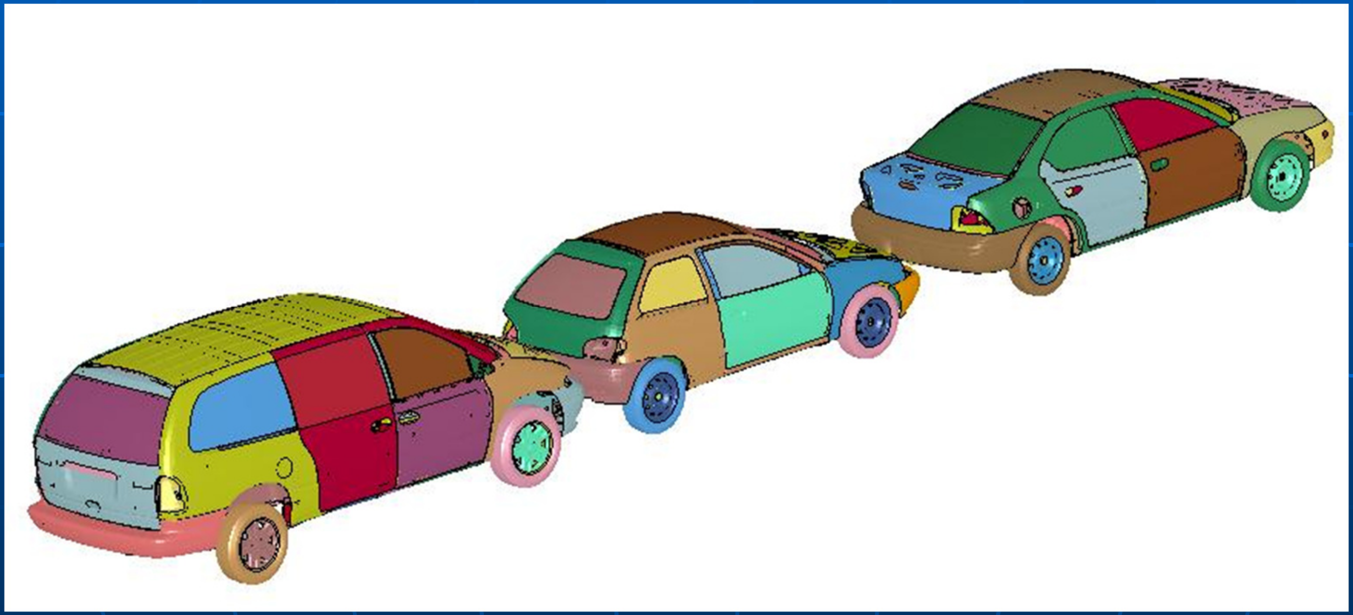
  - For a cluster the communication time is basically the time required for messages passing through the interconnection [Lin et al, 2000]

- Different types of interconnects
  - 100 BASET (TCP/IP) (2009: less used)
  - Gegi (TCP/IP) (2009: less used)
  - InfiniBand (OFED drivers) (Good and popular)

Effects of Interconnects

3 cars Benchmark Test

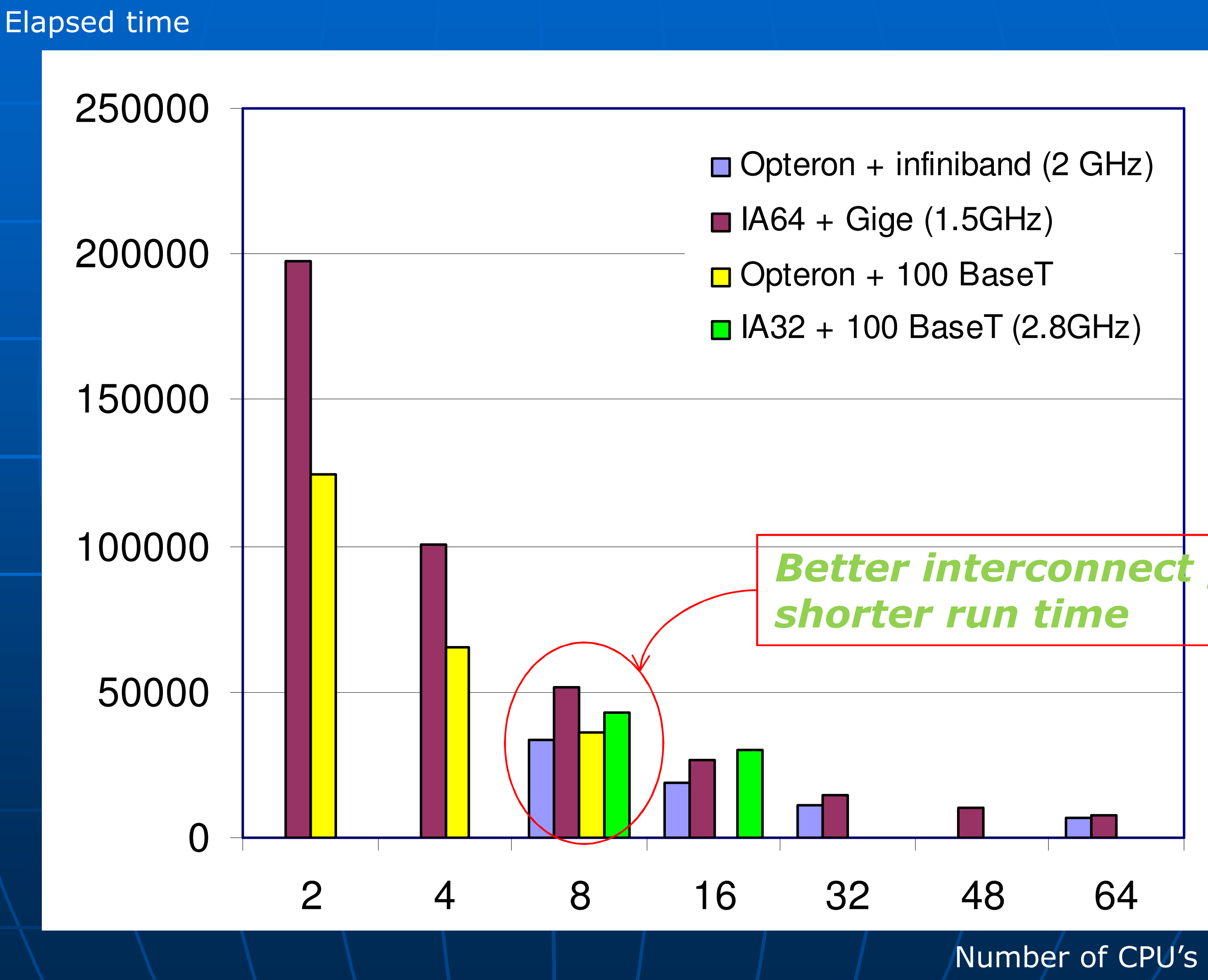
- Effect for the Benchmark test called 3 car Model. More on the model in the “Benchmark Test” Section.



794776 Elements and 1046 parts.

Effects of Interconnects

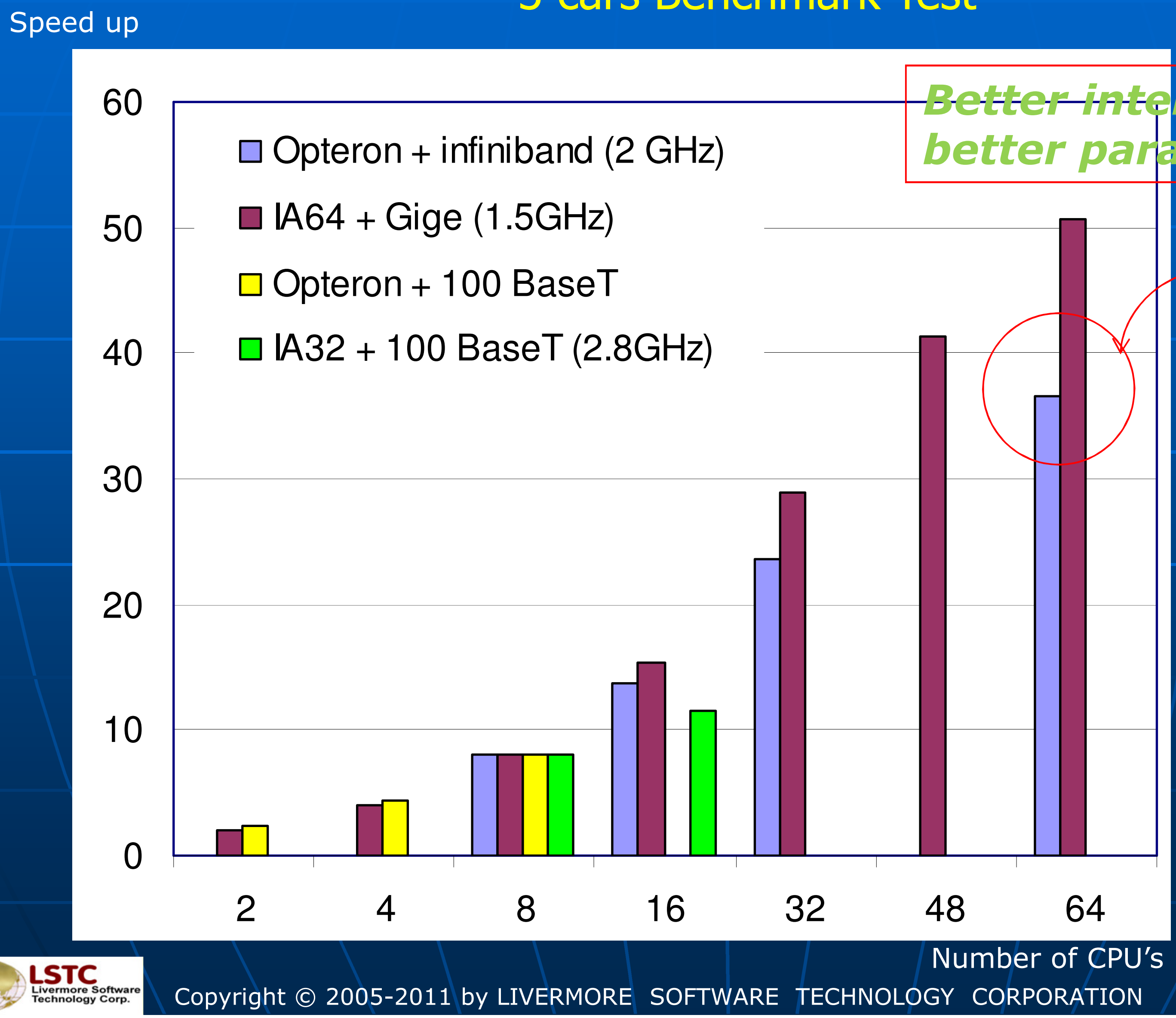
3 cars Benchmark Test



Better interconnect gives shorter run time

Effects of Interconnects

3 cars Benchmark Test



Better interconnect gives better parallel efficiency

Distribution of CPU time

In order to investigate the scaling of different phases of the MPP run [Zhu, 2005] made runs with the Neon (and the 3 car) benchmark test. She looked at 3 different phases of the run:

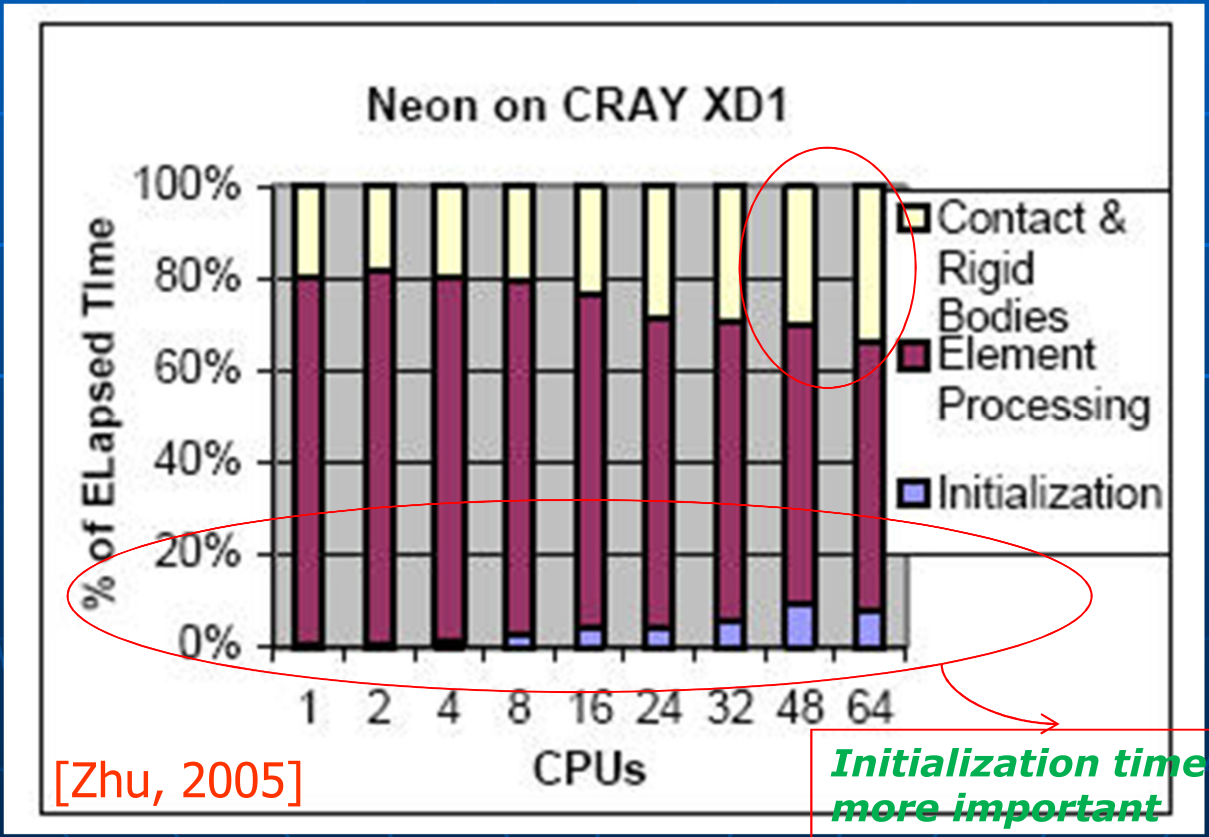
- Initialization: The time spend on reading the deck, allocate memory, domain composition does not scale since this is done serial on 1 CPU. However, the time is relative small.

- Element Processing: The phase for element processing i.e., calculation of motion, forces, stresses etc. is scaleable and is one of the phases where most time is used.

- Contact and Rigid Bodies: The time spend in contact can also be significant depending of the problem. Both the contact and the rigid body routines are scaleable.

Distribution of CPU time

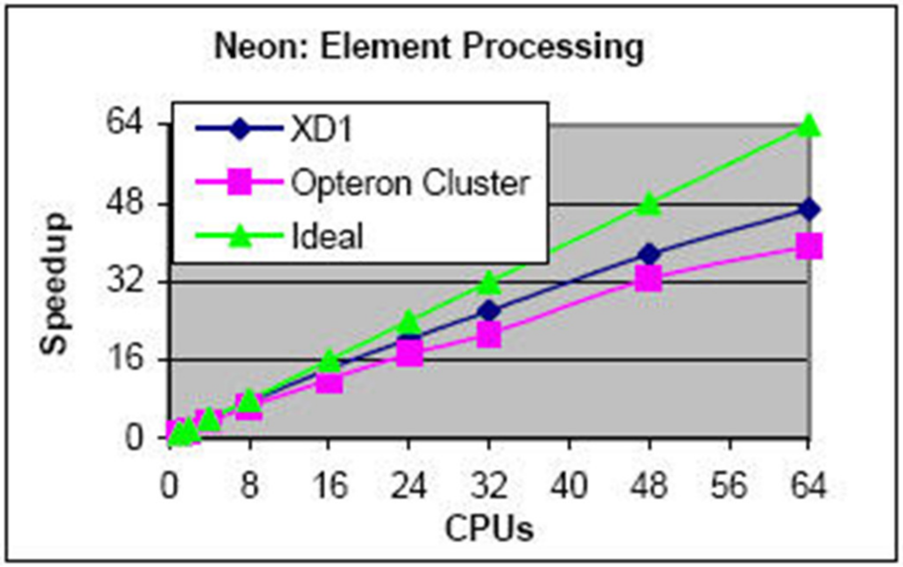
- The figure shows that the time spend in initialization and contact & rigid body routines are increased relatively to the time spend for element processing. These routines shows limited scaling for the specific model.

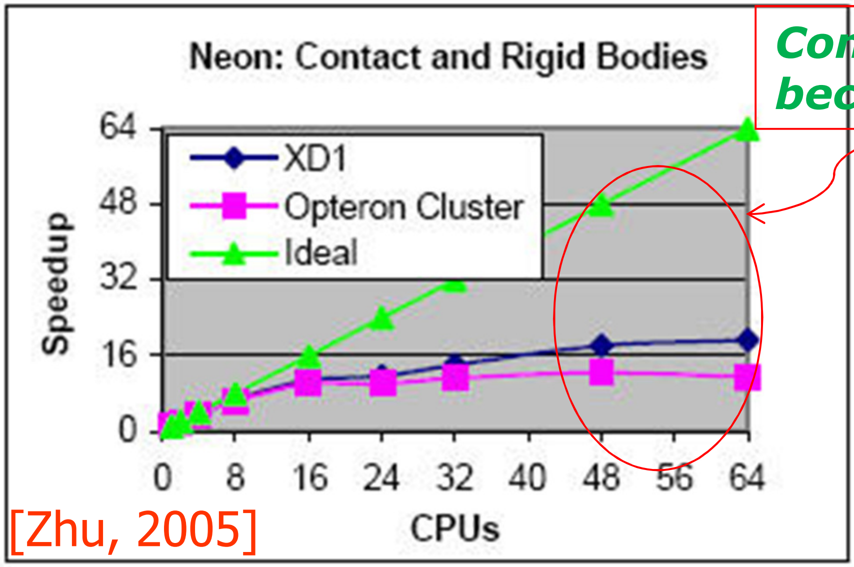


[Zhu, 2005]

Initialization time becomes more important

Distribution of CPU time

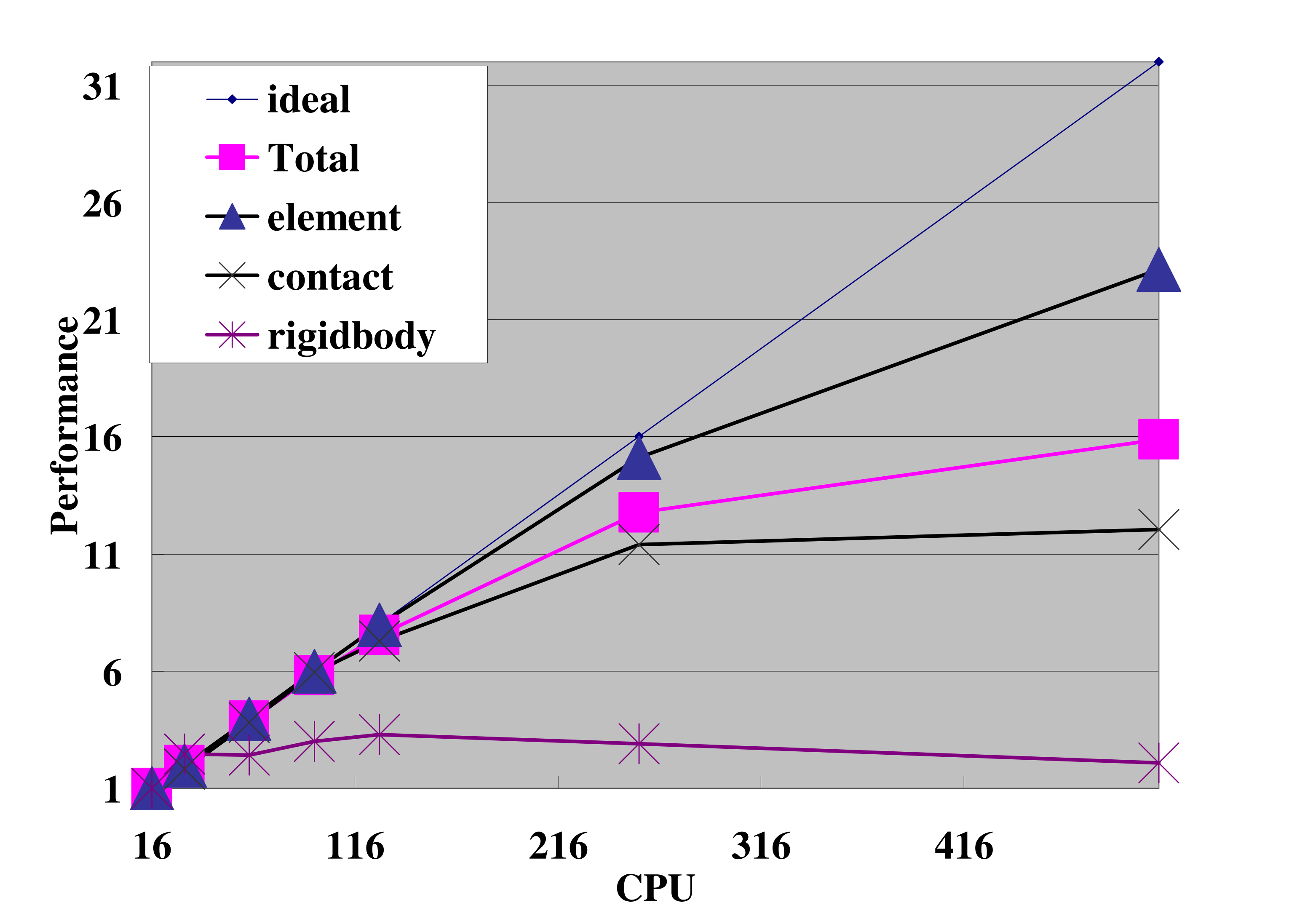




[Zhu, 2005]

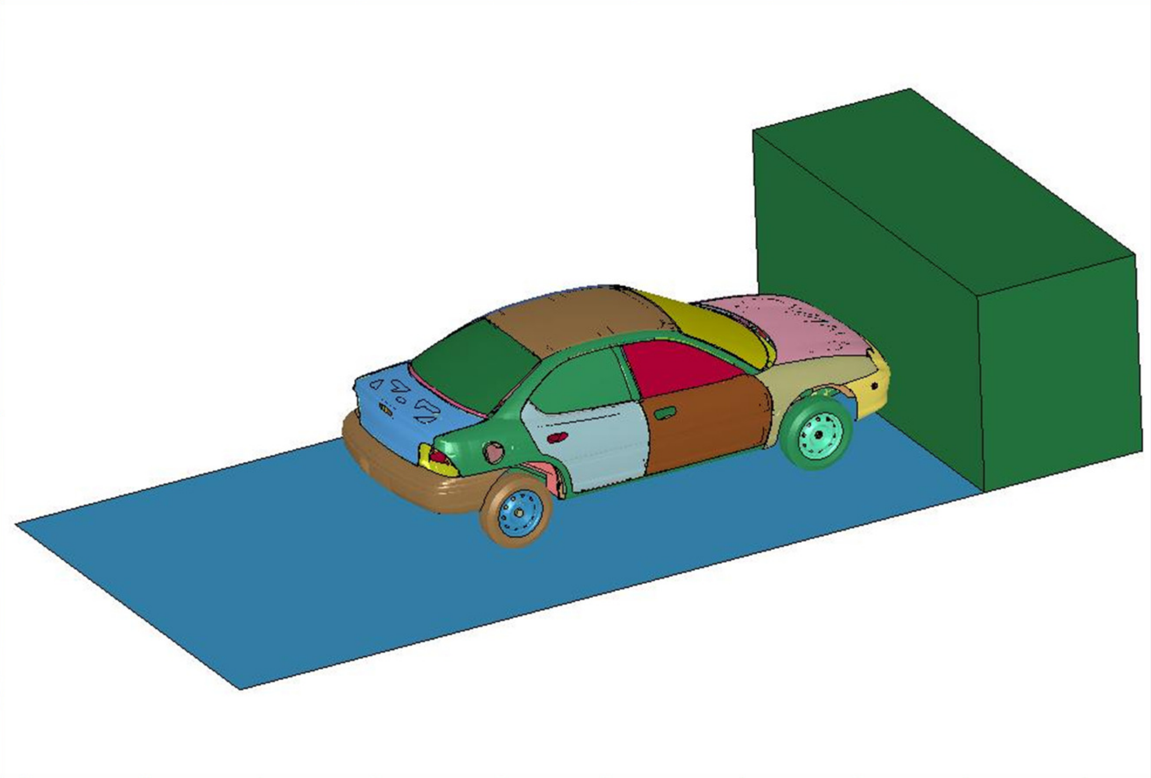
Contact load balancing becomes important

Distribution of CPU time



Effects of Decompositon

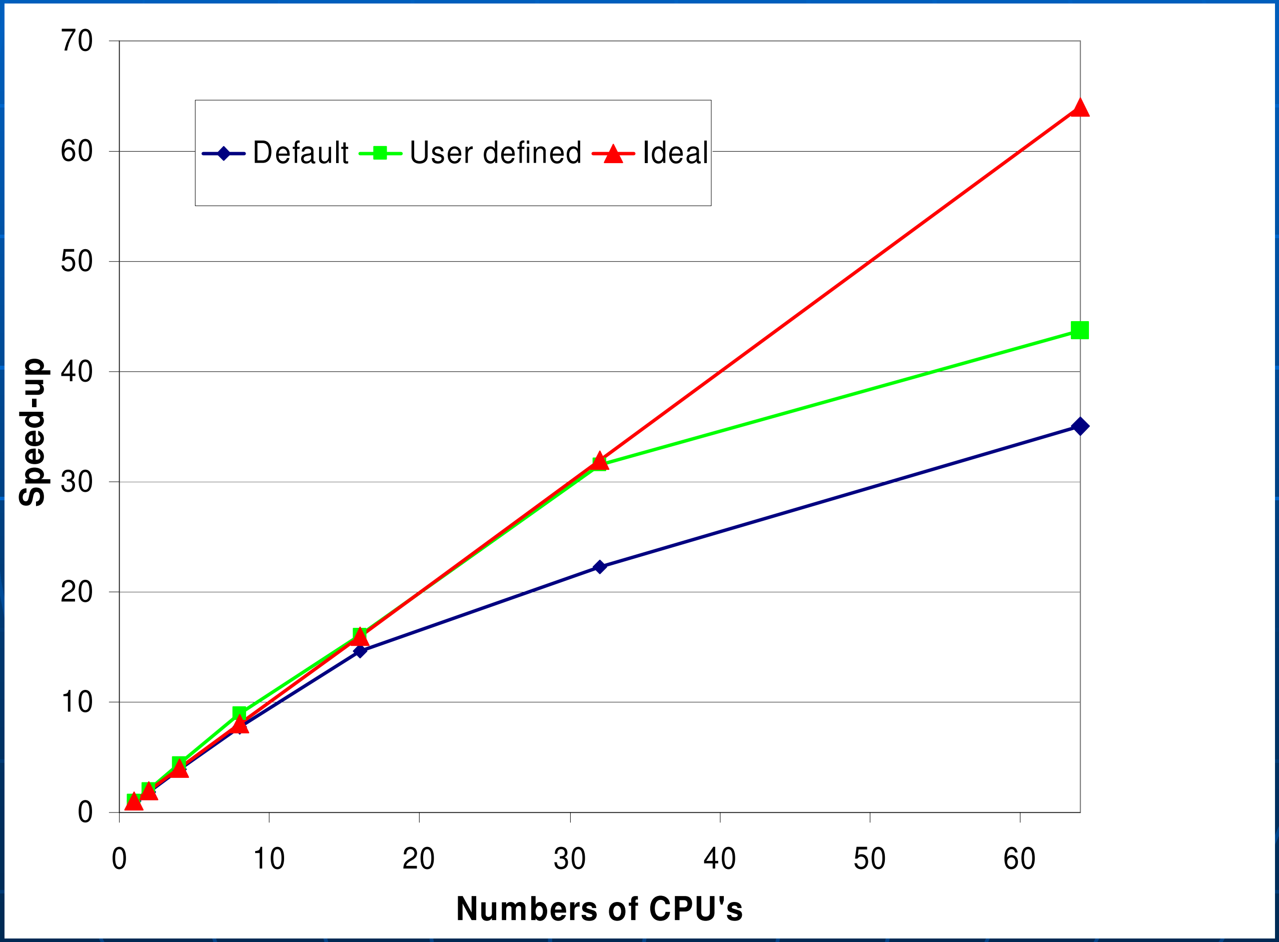
- Effect for the Benchmark test called Neon Model. The model consists of 267K elements, 30 millisecond frontal impact simulation. More on the model in the “Benchmark Test” Section.



535068 Elements and 322 parts.

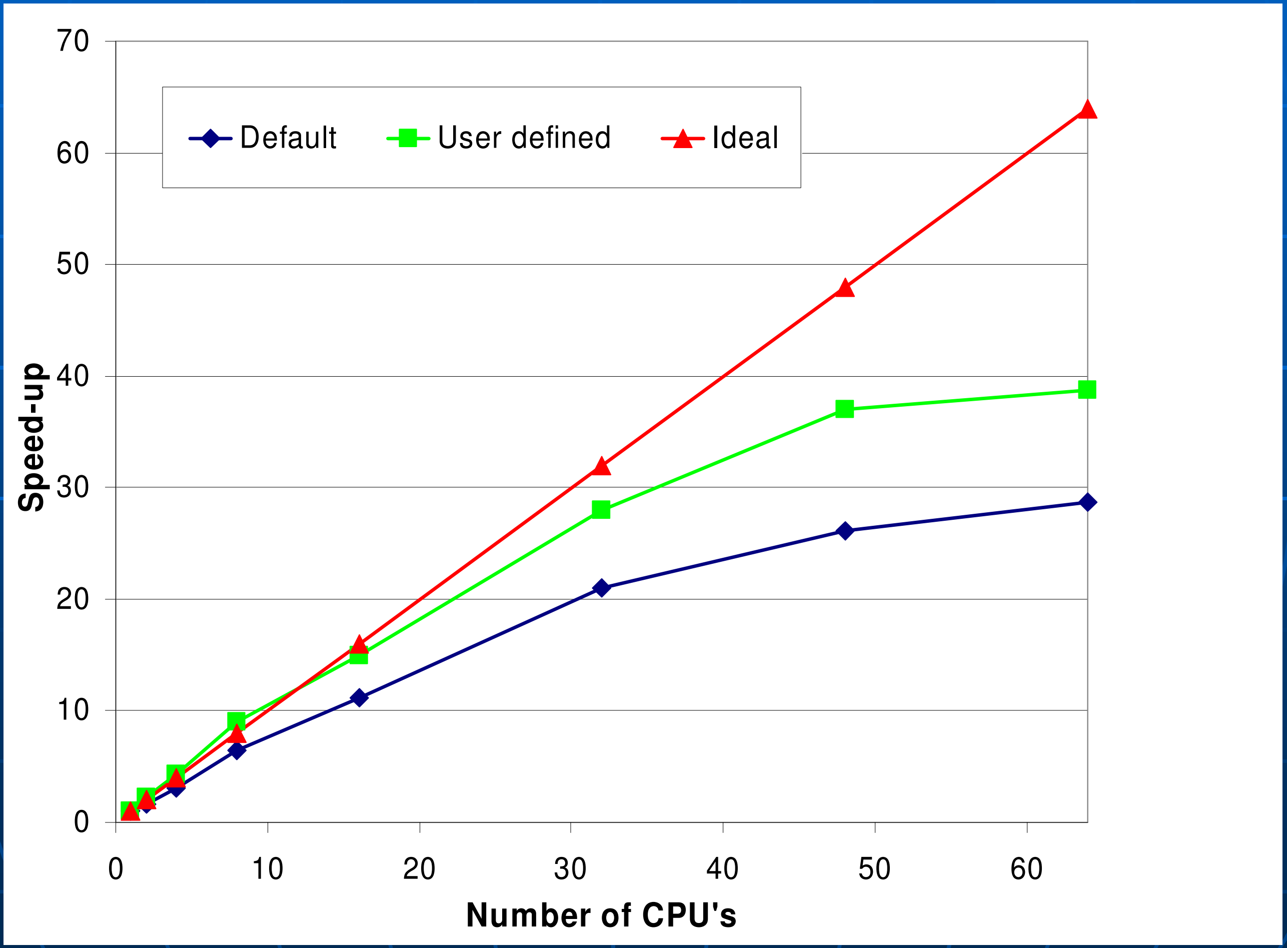
Effects of Decomposition

- The data plotted are based on the work published in [Chu et. al, 2000]. SGI machine running 30msec simulation.

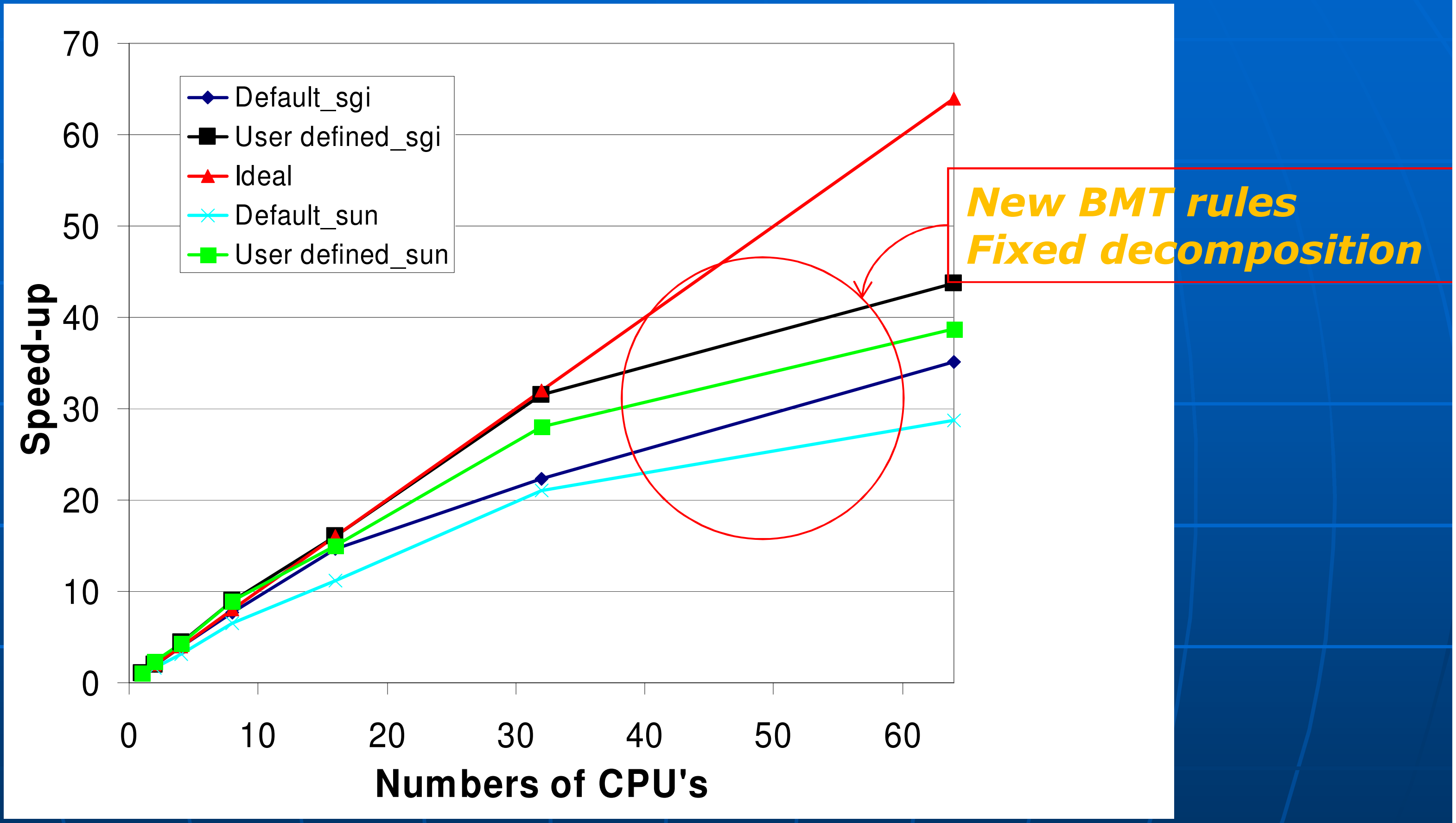


Effects of Decomposition

- The data plotted are based on the work published in [Roh, 2000]. Sun Machine running 10msec simulation.



Effects of Decomposition



New BMT rules
Fixed decomposition

- Be careful with performance conclusions between platforms ! Different termination time, memory, interconnections, version of the code etc.

Summary

- During the years LSTC has tested many different set-up for MPP. As shown there are many potential parameters that influence the scaling of the MPP code. Some of the most important ones are:
  - Decomposition (user controlled)
  - Memory/Cache System
  - Interconnections
  - MPI (2009: more or less same performance)
  - Compiler

Setup benchmark rule !!!!

Special Decomposition

Special Decomposition

- Introduction
- Load Balancing
- General Options for MPP
- Case Study
  - Crash
  - Metal Forming
  - ALE
- General Guidelines

Introduction

- Decomposition splits up the model in domains, which are done by the primary processor. Ideally the computational cost for each domain should be the same. Then there is an equal load balance.

- There are many factors affect the parallel performance
  - Boundaries of the generated domains.
  - Contact definitions
  - Special features used in the modeling

- The default decomposition used in the code is RCB (Recursive Coordinate Bisection)
  - RCB divides the model in half, each time slicing the current piece of the model perpendicular to one of the three axes
  - The axis along which the current piece of the model is longest is chosen
  - The method tends to generate cube shaped domains aligned along the coordinate axes

Introduction

- The user decomposition can only control through the p-file in the early releases. It can be included in the keyword commands (***CONTROL_MPP_*****_option_**) from 970. If the same option is appeared in both input, the option in the pfile has the higher priority. There are four sections: *Directory, Decomposition, Contact and General*. Each section has relevant commands, see Appendix O.

- One processor is doing the decomposition, which can require a large amount of memory, more than necessary in the simulation.
  - Therefore, there are two memory options on the command line when executing LS-DYNA® MPP:

```
mpirun –np 64 mpp971 i=test.k memory=80m memory2=20m  p=pfile
```

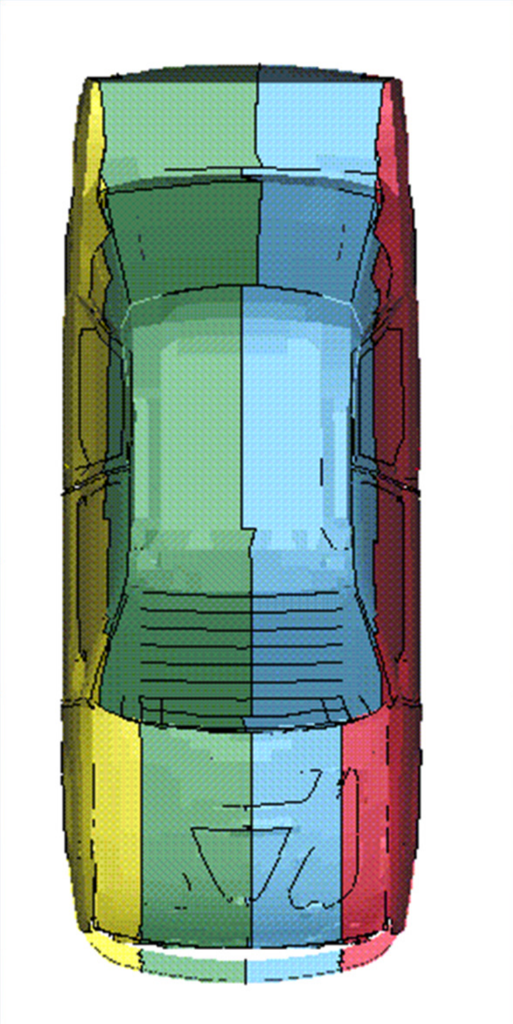
memory is for processor 0 for decomposition and simulation. memory2 is for the simulation for the rest of processors

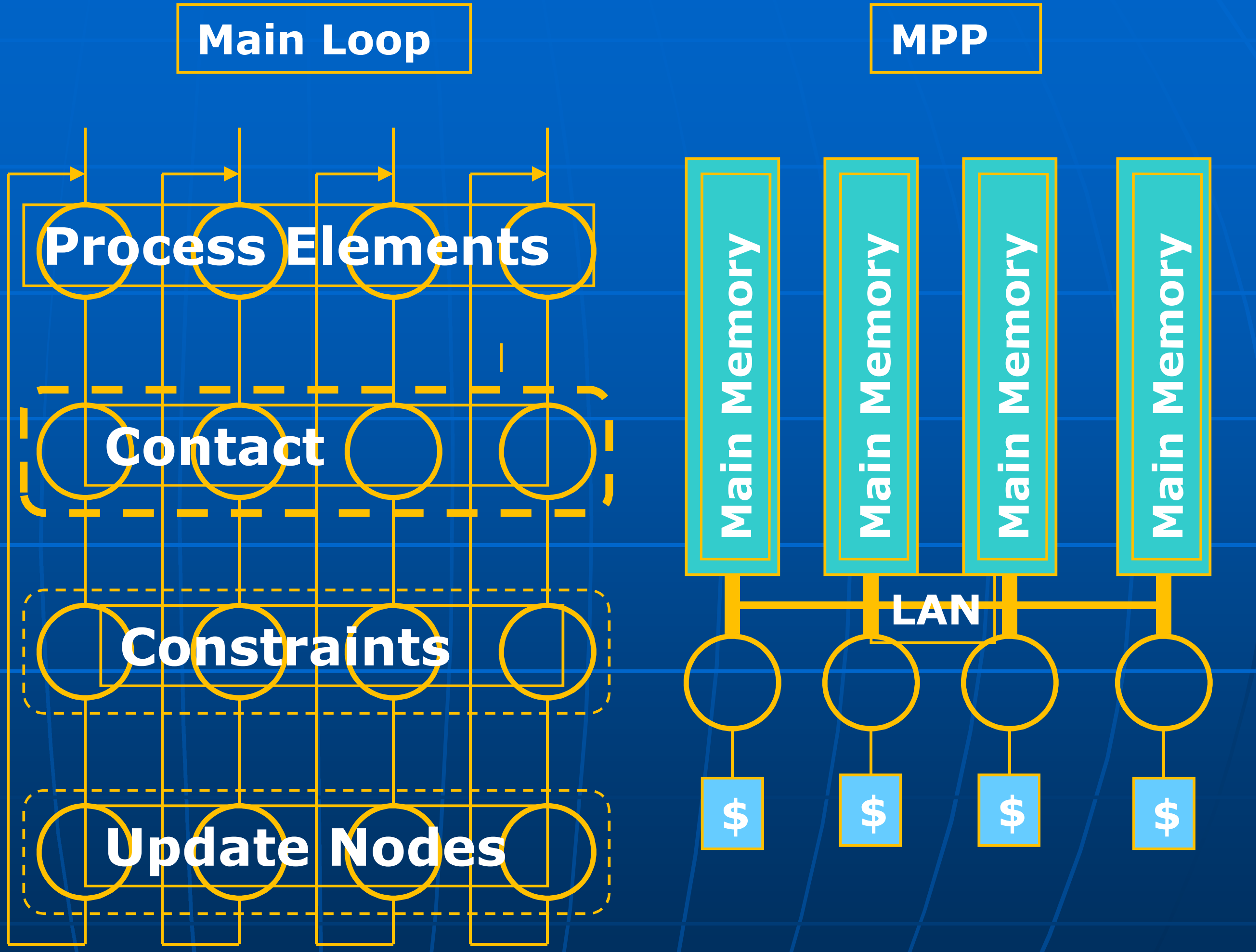
  - Performing multiple steps run
    1. Get keyword translated to structure input
    2. Use structure input to get pre-decomposition file
    3. Restart job with pre-decomposition file

Load Balancing

- Decomposition method
  - Recursive Coordinates Bisection (default)
- Distrorted subdomain
  - Contact or coupling definitions (major)
  - Different element formulation (minor)
  - Force summation over shared nodes (minor)

Load Balancing





Main Loop

Process Elements

Contact

Constraints

Update Nodes

MPP

Main Memory

Main Memory

Main Memory

Main Memory

LAN

$

$

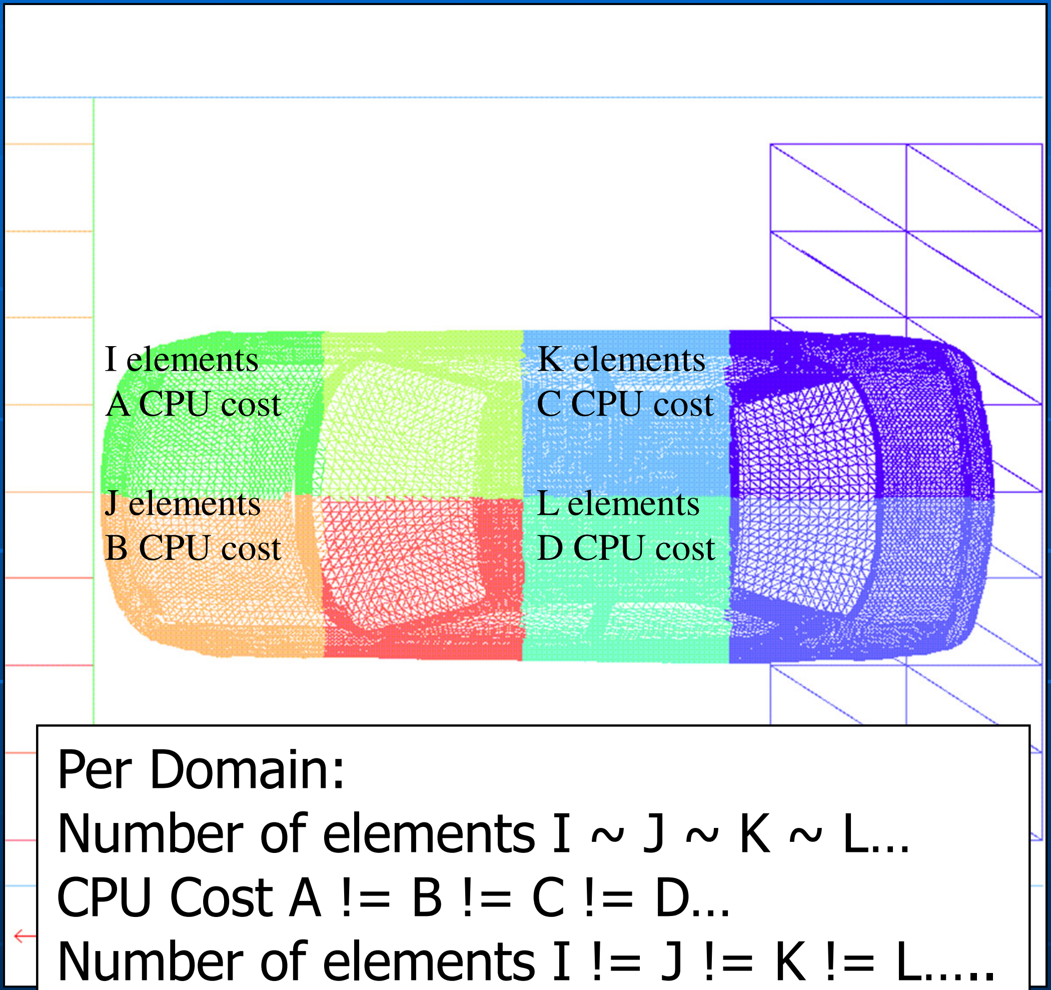
$

$

Load Balancing

(a) Element Cost

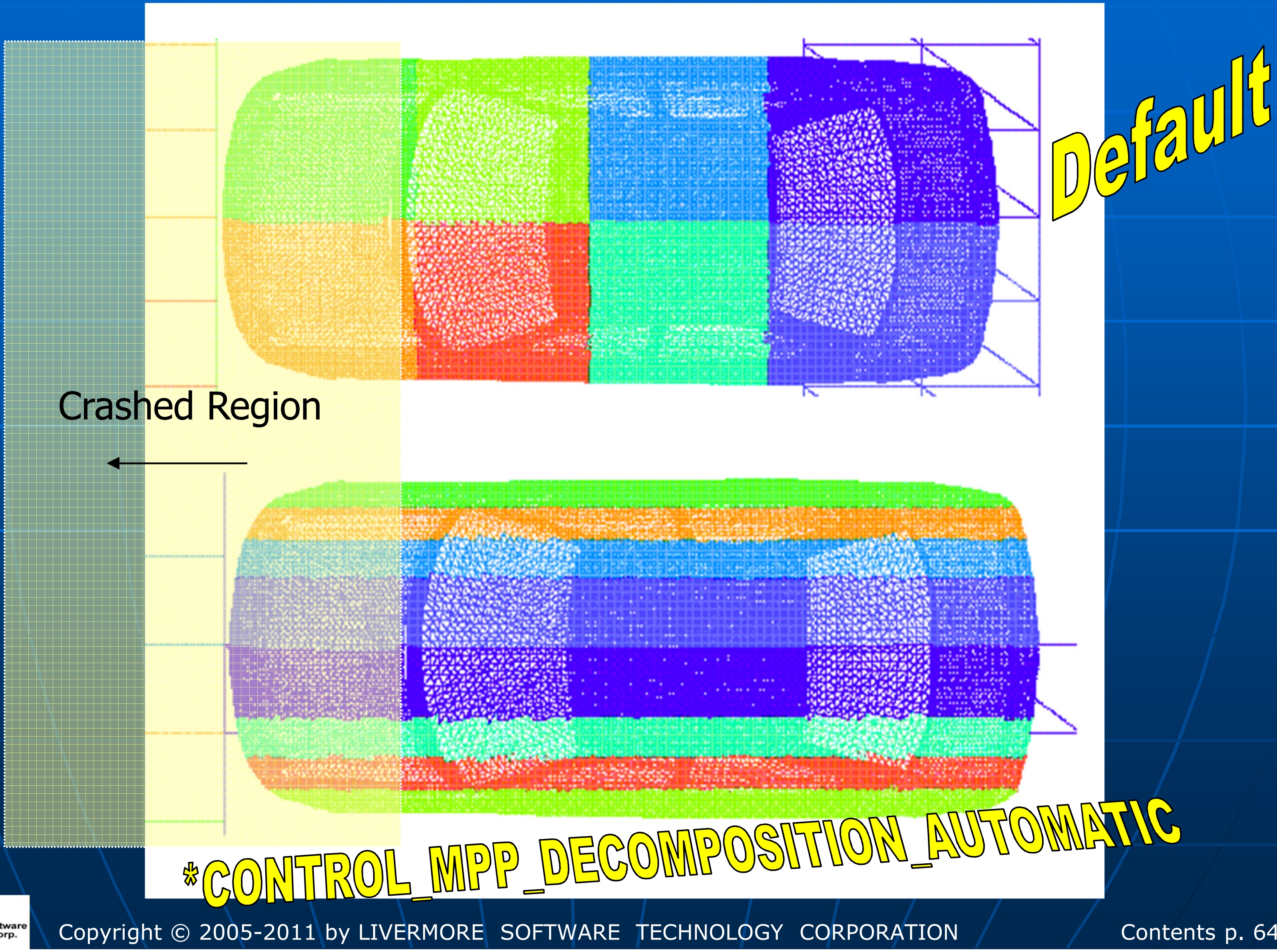
The Domains are based on element cost not number of elements



Per Domain:
Number of elements I ~ J ~ K ~ L…
CPU Cost A != B != C != D…
Number of elements I != J != K != L…..
CPU Cost A ~ B ~ C ~ D….

Load Balancing

(b) Contact Cost



Load Balancing

information during execution

```
host1
29593 jason   15   0  190M 190M  6164 R      79.2  4.8  1476m mpp970
29586 jason    9   0  404M 404M  6960 S       6.7 10.3 125:38 mpp970
host2
 7599 jason   18   0  178M 178M  6104 S      10.2  4.5 178:25 mpp970
 7590 jason   10   0  170M 170M  5828 S       3.6  4.3  84:47 mpp970
host3
20275 jason   18   0  186M 185M  6072 R      54.8  4.7  1019m mpp970
20284 jason    9   0  166M 166M  5936 S       1.5  4.2  44:04 mpp970
host4
20849 jason   13   0  169M 169M  5884 S      16.8  4.3  56:09 mpp970
20858 jason   12   0  167M 167M  5824 S      12.8  4.2 102:27 mpp970
```

Load Balancing

information after execution

```
mes0000
 Element processing ... 3.4474E+02   57.61     6.7254E+02   47.54
 -----
 Contact algorithm .... 1.4906E+02   24.91     4.2288E+02   29.89
   Interface ID       1 1.4536E+02   24.29     4.1547E+02   29.37

mes0001
 Element processing ... 2.9436E+02   52.75     6.5738E+02   46.46
 Contact algorithm .... 2.2382E+02   40.11     4.5323E+02   32.03
   Interface ID       1 2.1671E+02   38.84     4.2008E+02   29.69
   Interface ID      20 2.2295E+00    0.40     1.0072E+01    0.71
   Interface ID      21 1.4300E+00    0.26     1.0603E+01    0.75

mes0002
 Element processing ... 2.7035E+02   50.00     6.7720E+02   47.86
 Contact algorithm .... 2.3439E+02   43.35     4.5477E+02   32.14
   Interface ID       1 2.1606E+02   39.96     4.1339E+02   29.21
   Interface ID      20 7.2402E+00    1.34     2.2589E+01    1.60
   Interface ID      21 6.2605E+00    1.16     1.0594E+01    0.75
-----
```

General Options for MPP

P-file

```
directory { global  tempdir local /torch2/nmeng/tempdir }
decomposition { C2R 0 0 0 0 0 1 1 0 0 sy 1000 show }
contact { bucket 100 }
general { nodump }
```

- The *p-file* is case insensitive and have a free format input.
- Words and brackets must have either a space, tab or a newline character on each side.
- Consists of four sections: ***directory, decomposition, contact and general***

General Options for MPP

directory

The directory option holds directory specific options

- global *path*

Path to a directory accessible to all processors. This directory will be created if necessary. Default = current working directory

- local *path*

Path to a processor specific local directory for scratch/local files. This directory will be created if necessary. This is of primary use on systems where each processor has a local disk attached to it. Default = global path

- rep path

- transfer_files

Move output files back from local disk to starting directory or move restart files from starting directory to target local disk

General Options for MPP

decomposition

- rx ry rz sx sy sz c2r s2r 3vec mat
  See the section Decompositions for details about these decomposition options.

- rcblog filename
  This option is ignored unless the decomposition method is RCB. If the indicated file does not exist, then a record is stored of the steps taken during decomposition. If the file exists, then this record is read and applied to the current model during decomposition. This results in a decomposition as similar as possible between the two runs. For example, suppose a simulation is run twice, but the second time with a slightly different mesh. Because of the different meshes the problems will be distributed differently between the processors, resulting in slightly different answers due to roundoff errors. If an rcblog is used, then the resulting decompositions would be as similar as possible.

General Options for MPP

decomposition

- slist n1,n2,n3,...

  This option changes the behavior of the decomposition in the following way. n1,n2,n3 must be a list of sliding interfaces occurring in the model (numbered according to the order in which they appear, starting with 1) delimited by commas and containing no spaces (eg "1,2,3" but not "1, 2, 3"). Then all elements belonging to the first interface listed will be distributed across all the processors. Next, elements belonging to the second listed interface will be distributed among all processors, and so on, until the remaining elements in the problem are distributed among the processors. Up to 5 interfaces can be listed. It is generally recommended that at most 1 or 2 interfaces be listed, and then only if they contribute substantially to the total computational cost. Use of this option can increase speed due to improved load balance.

- sidist n1,n2,n3,...

  This is the opposite of the silist option: the indicated sliding interfaces are each forced to lie wholly on a single processor (perhaps a different one for each interface). This can improve speed for very small interfaces by reducing sychronization between the processors.

General Options for MPP

general

The general option holds general options.

- nodump

  If this keyword appears, all restart dump file writing will be suppressed

- nofull

  If this keyword appears, writing of d3full (full deck restart) files will be suppressed.

- nobeamout
- binoutonly
- Lstc_reduce

General Options for MPP

contact

The general option holds general options.

- groupable integer

  If this keyword appears, LS-DYNA/MPP will try to group type 3,5,10 contacts into one big communicator to save communication latency. Soft=2 contacts are not considered in this process.

General Options for MPP

To View the Decomposition (a)

```
mpirun –np 64 mpp_executable i=input p=pfile
```

```
general
decomp { show }
contact
directory
```

show : output the decomposition and stop

Or in the input deck:

```
*CONTROL_MPP_DECOMPOSITION_SHOW
```

General Options for MPP

To View the Decomposition (b)

```
mpirun –np 64 mpp_executable i=input p=pfile
```

```
decomp { outdecomp }
```

outdecomp : output the decomposition file and job keep running

This output file can be read back by lsprepost

lsprepost > view > MPP > load

General Options for MPP

There are many more options and correspondent *CONTROL_MPP keyword.

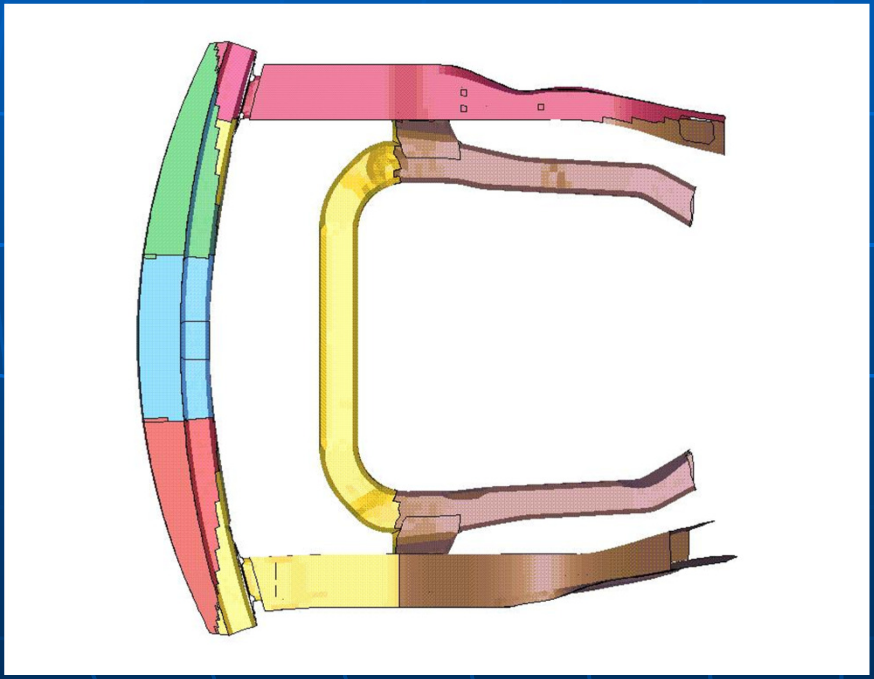
Please check the User’s Manual Appendix O

Case study

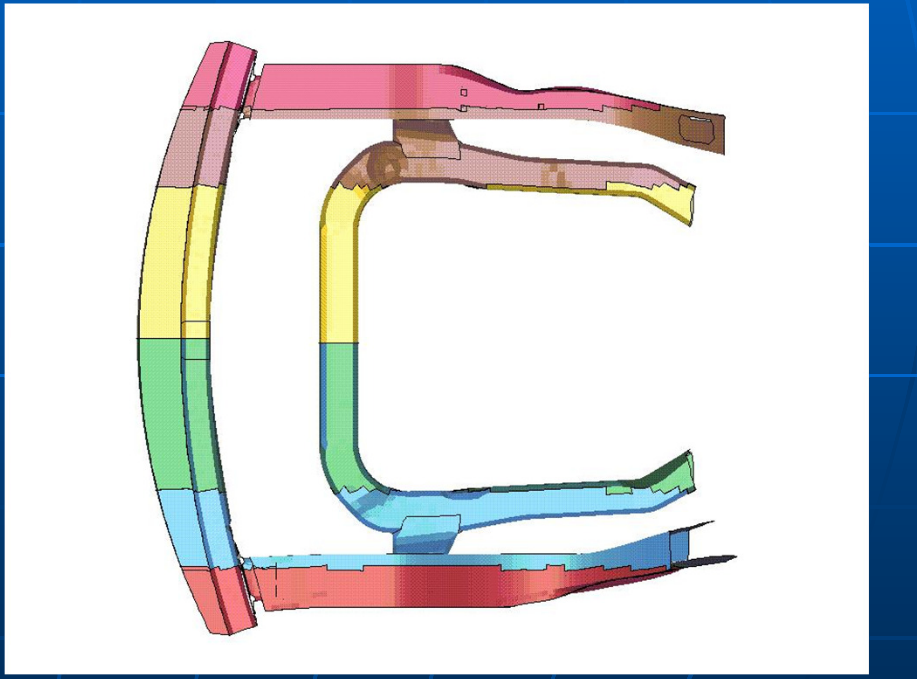
- Bumper Impact
- Side Impact
- ODB
- Metal Forming
- ALE Airbag Simulation

Case Study for Crash: Bumper

Default RCB

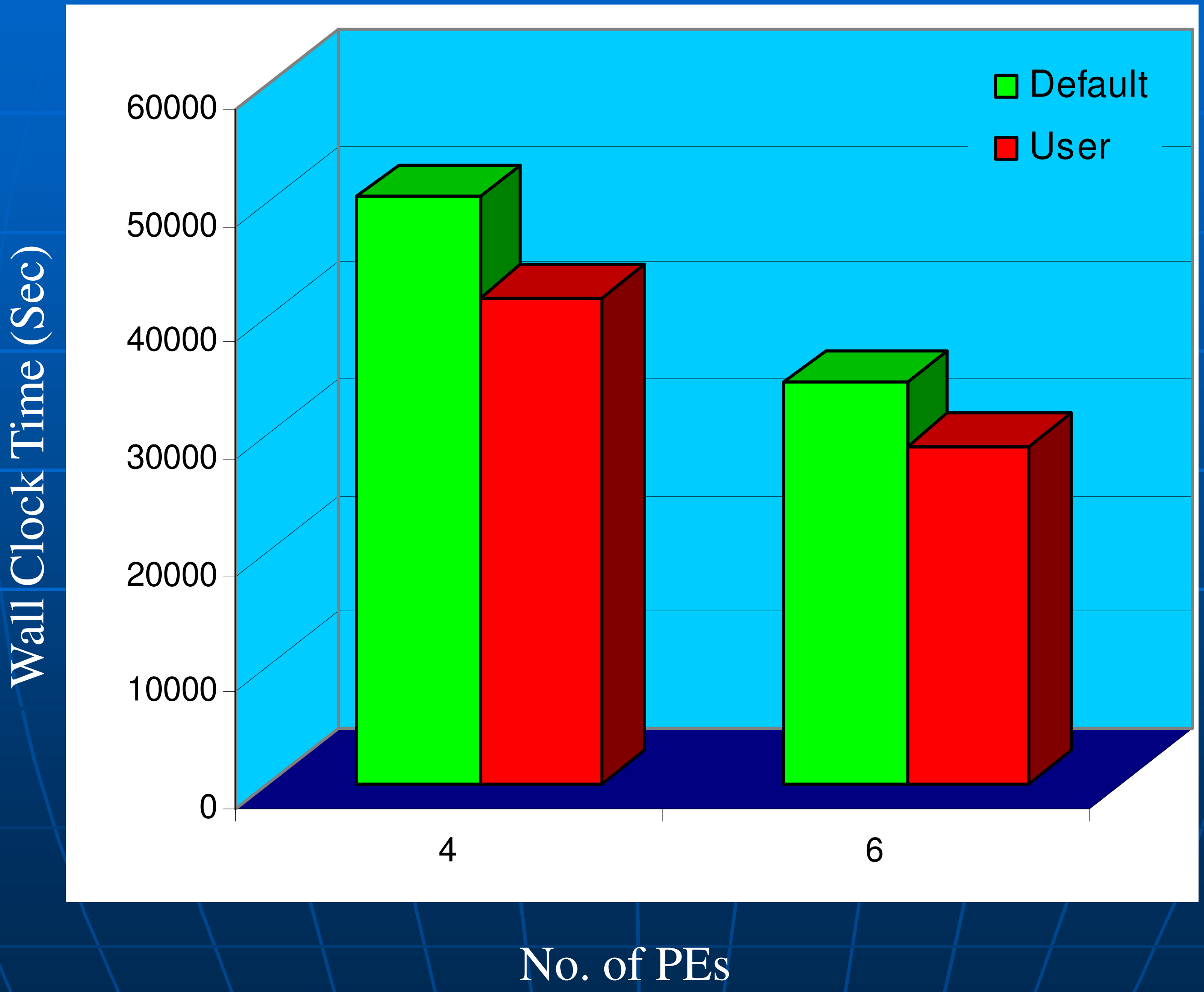


sy 5.0

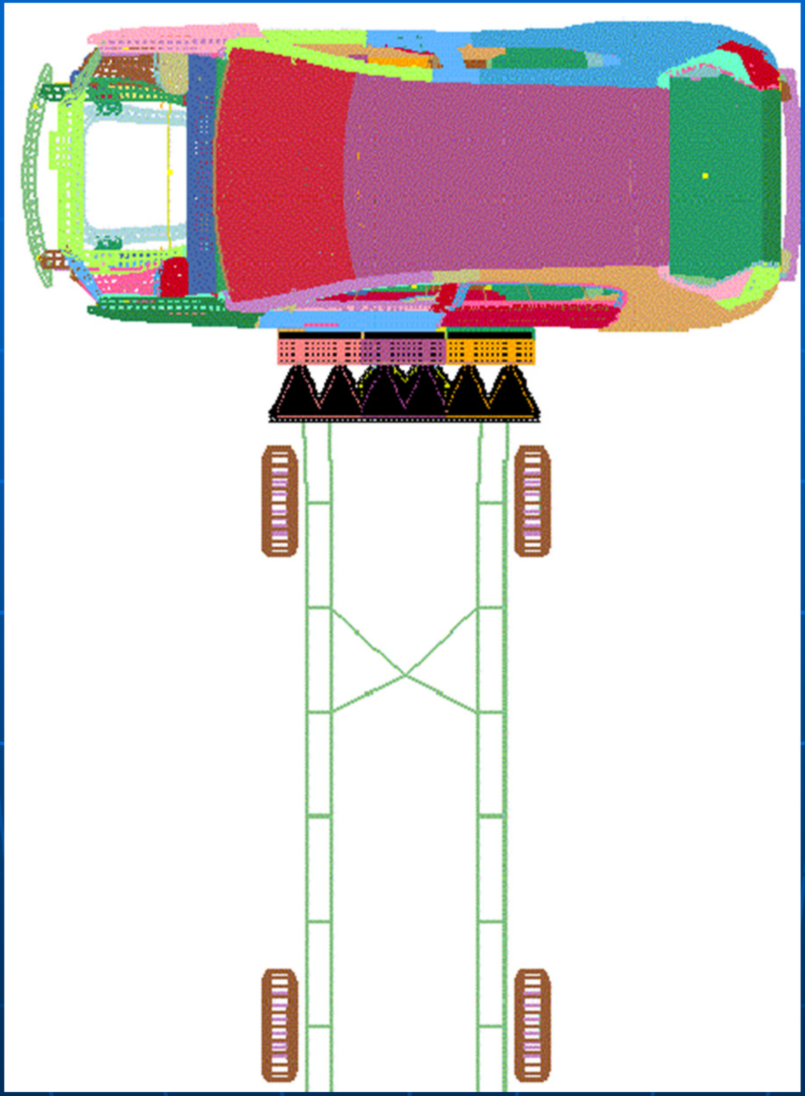


Case Study for Crash: Bumper

Performance Improvement via Changing Partition



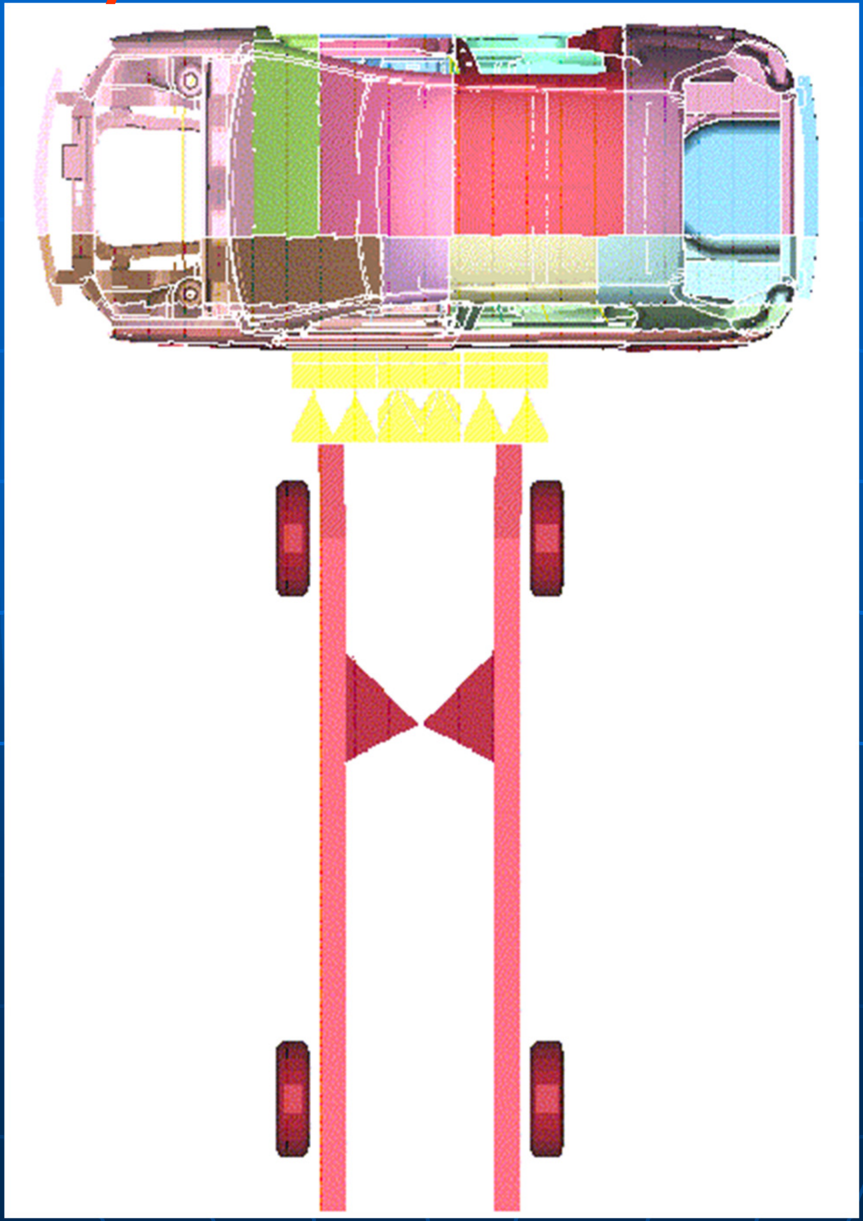
Case Study for Crash: Side Impact



13 contacts and 10,11,12,13 are around barrier and car

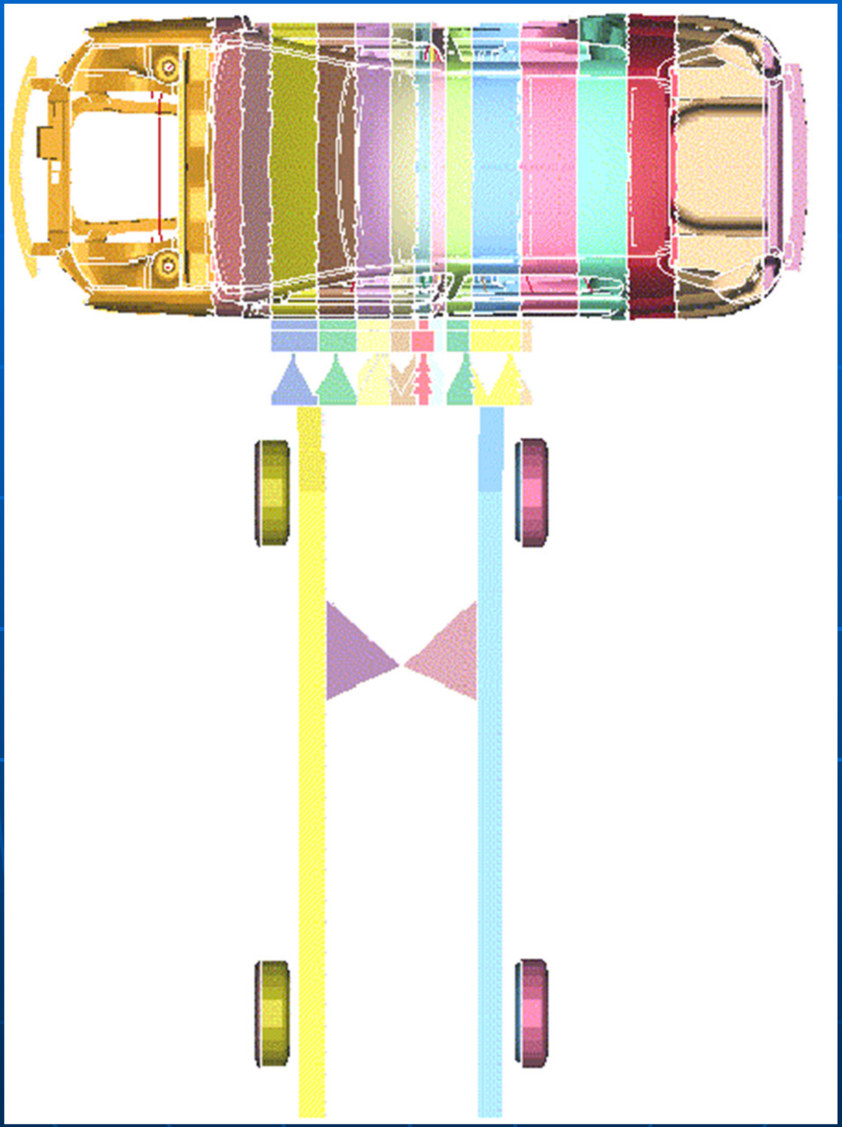
Case Study for Crash: Side Impact

Default



Case Study for Crash: Side Impact

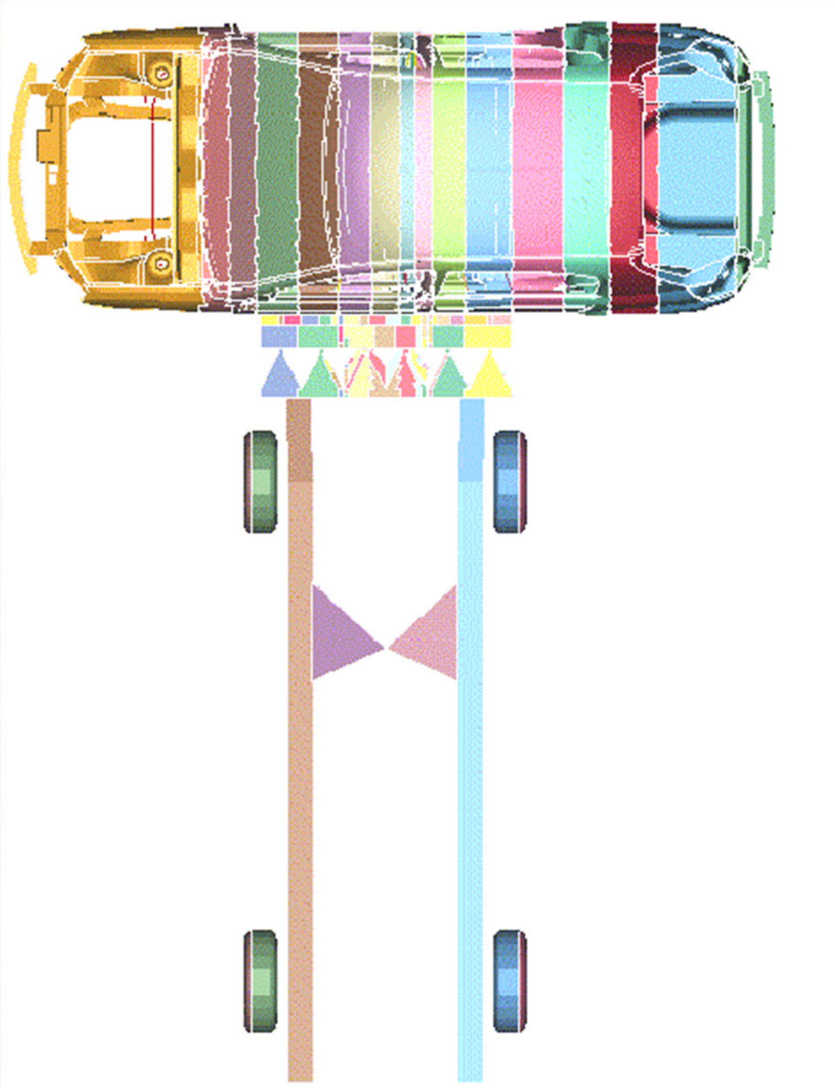
Method 1



Decomp { sx 1000 numproc 16 show }

Case Study for Crash: Side Impact

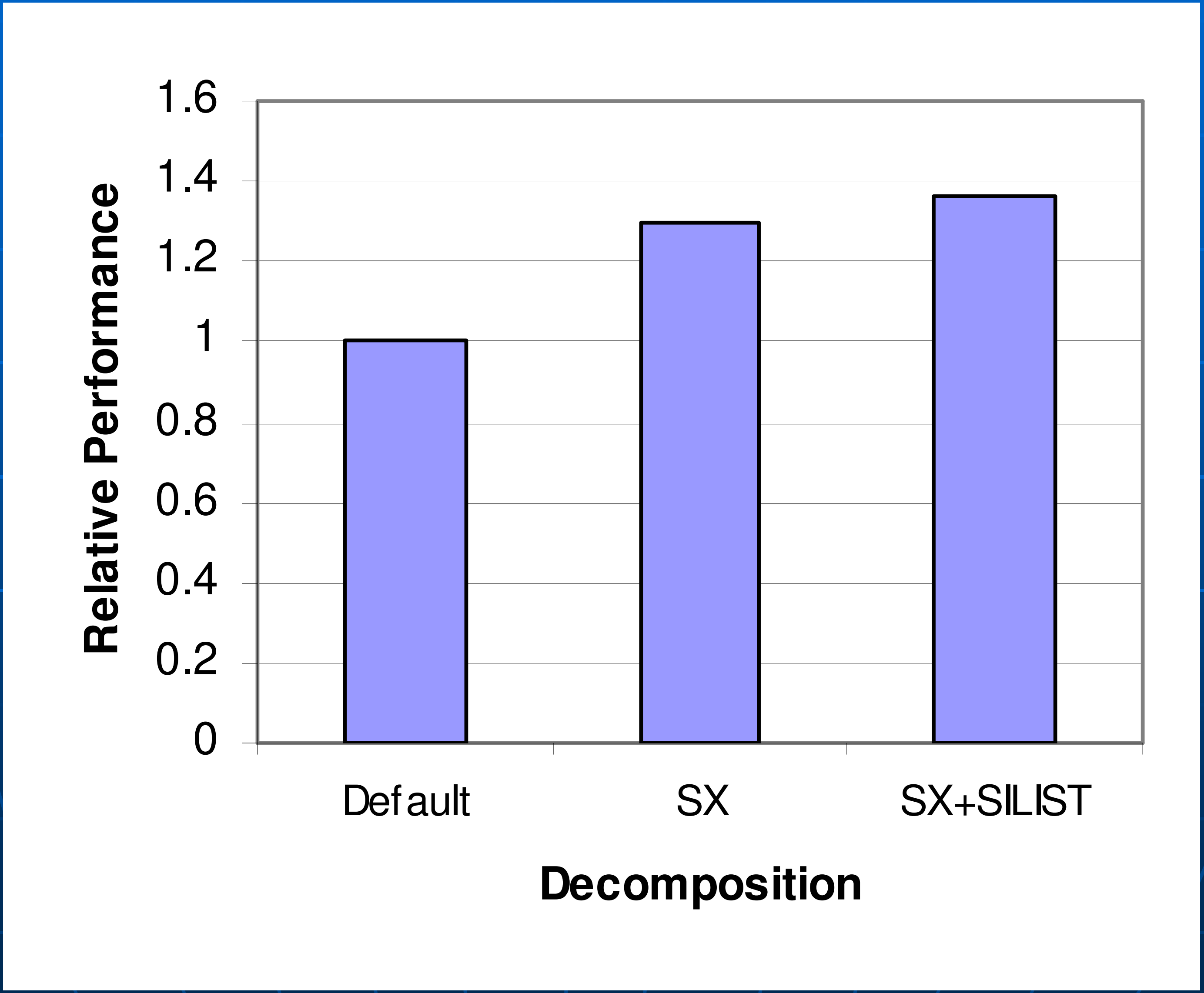
Method 2



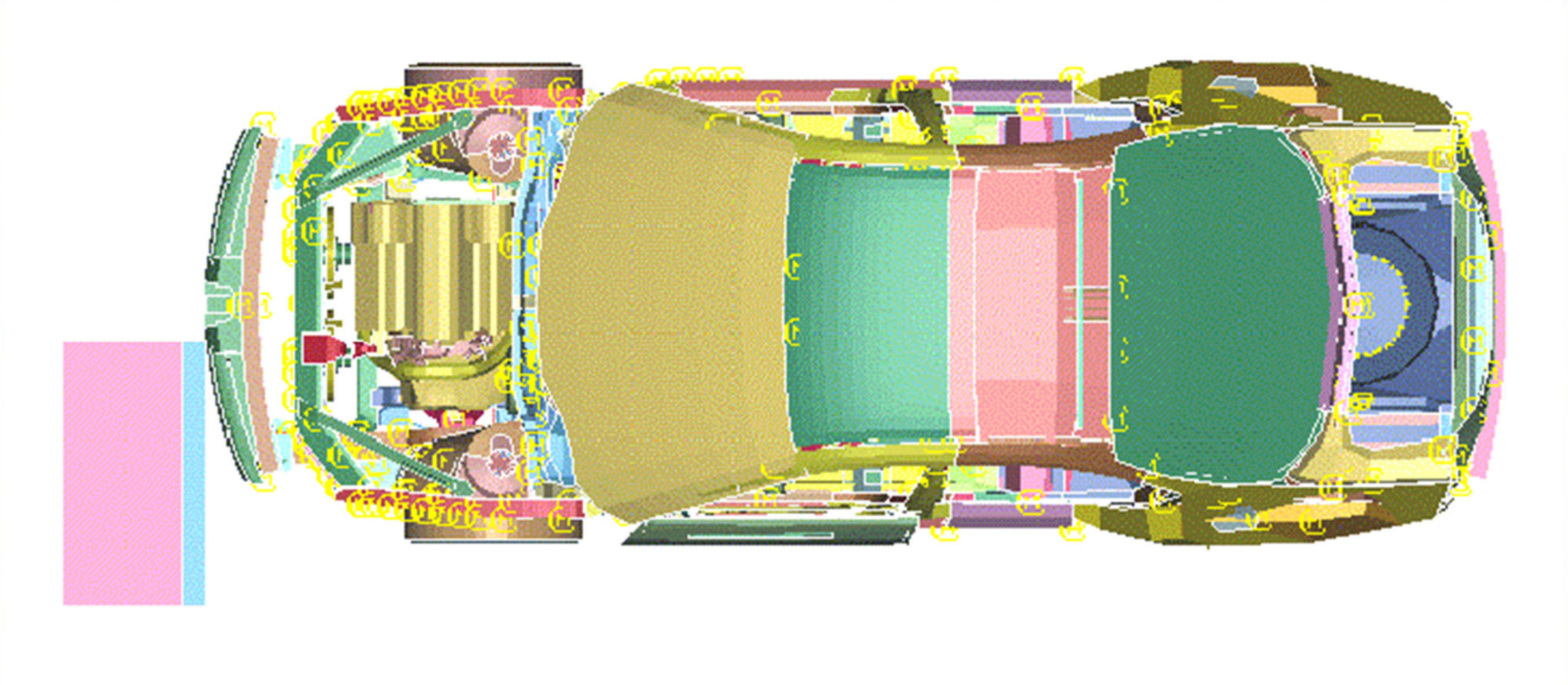
Decomp {sx 1000 silist 10,11,12,13 numproc 16 show }

Case Study for Crash: Side Impact

Timing Comparison first 5000 cycles, 8 CPU's



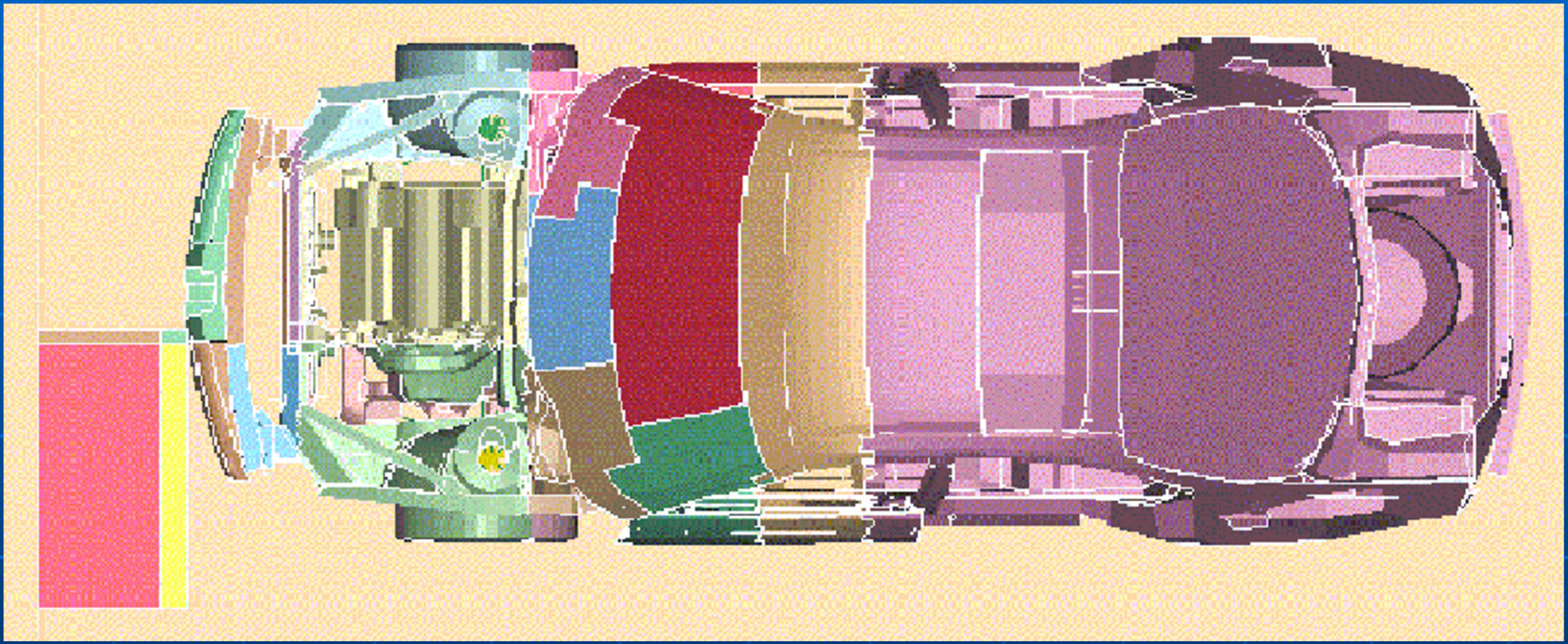
Case Study for Crash: ODB



One single surface contact

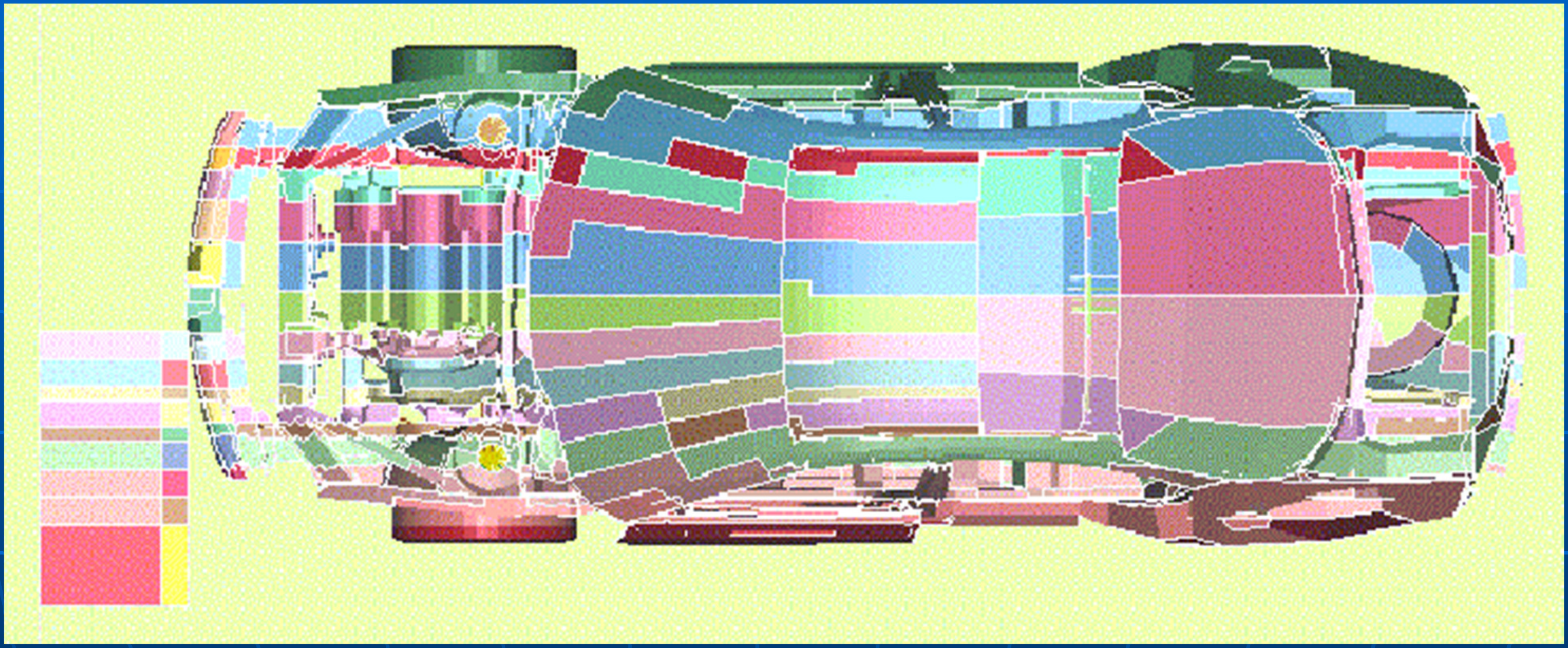
Case Study for Crash: ODB

Default



Case Study for Crash: ODB

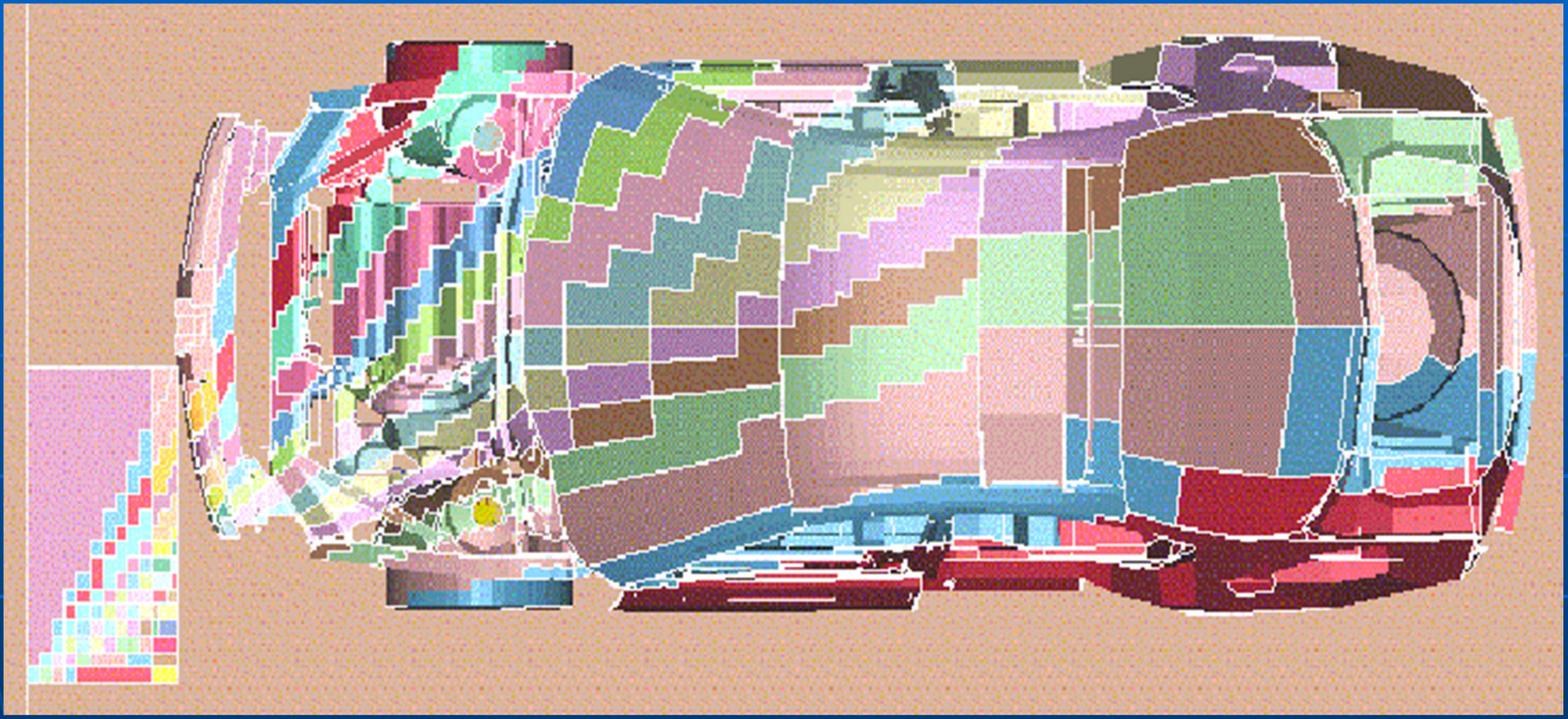
Method 1



Decomp { sy 1000 numproc 16 show }

Case Study for Crash: ODB

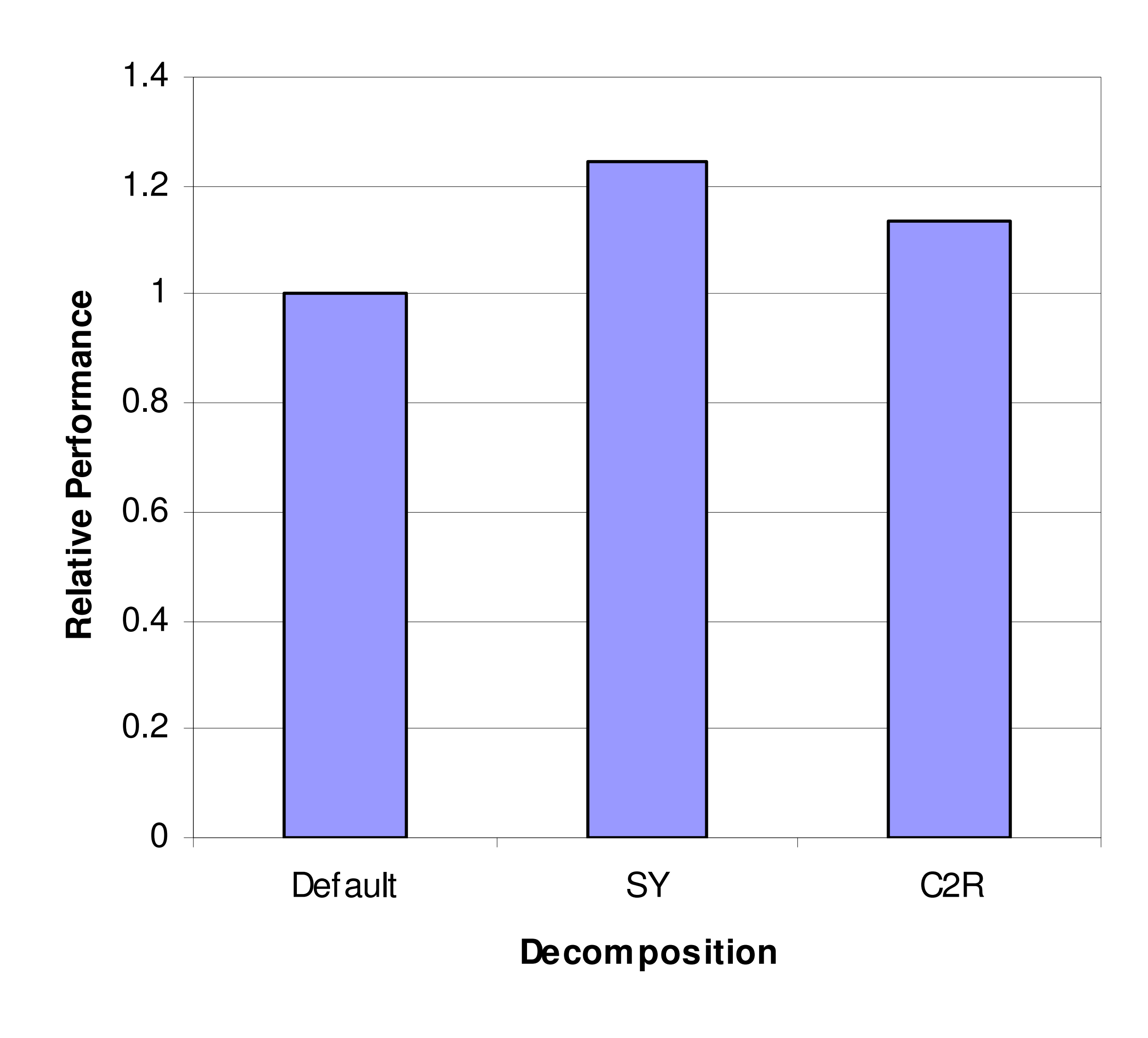
Method 2



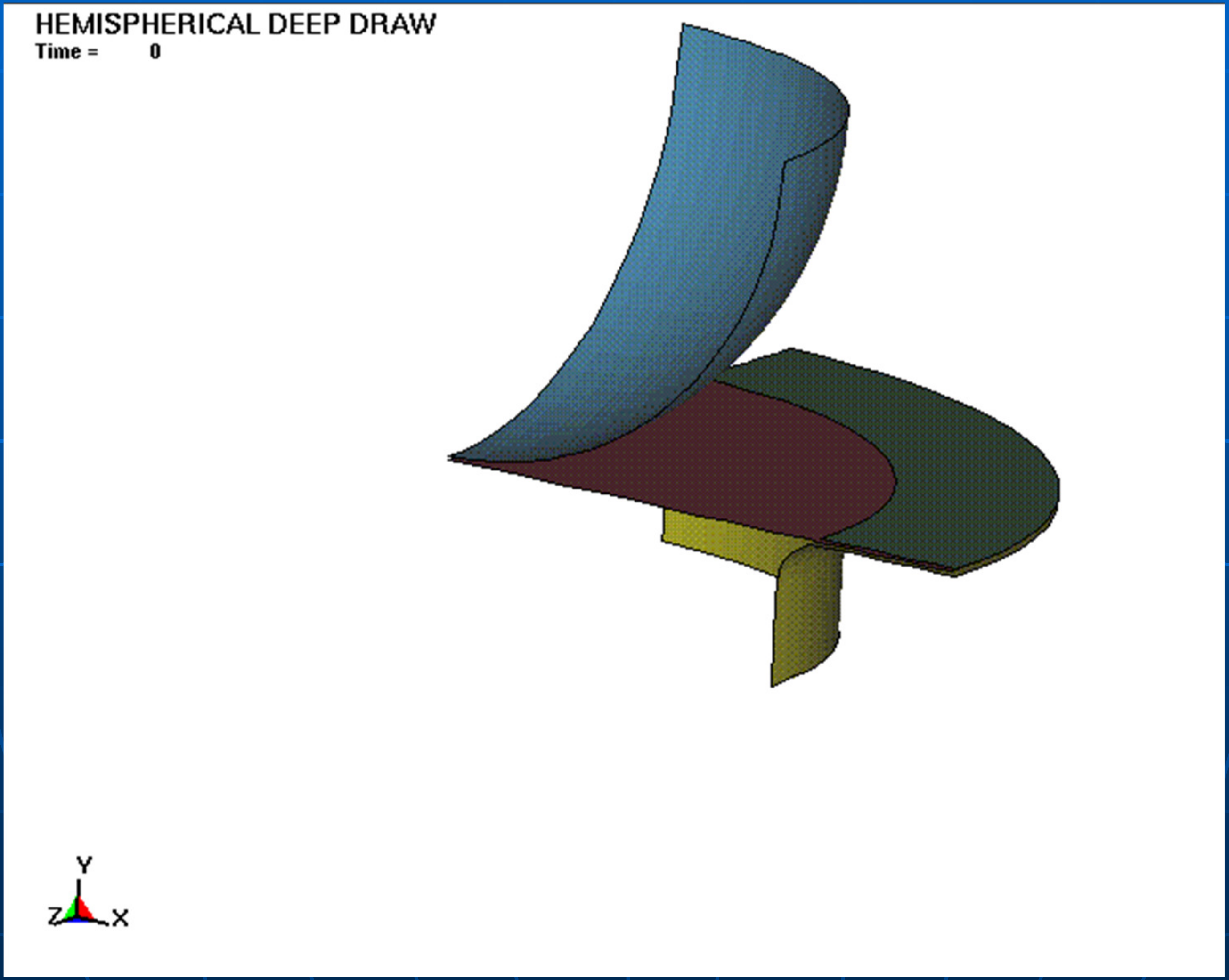
Decomp { C2R 177 –1134 1143 0 0 1 1 0 0 sy 10000 numproc 16 show }

Case Study for Crash: ODB

Timing Comparison first 5000 cycles, 8 CPU



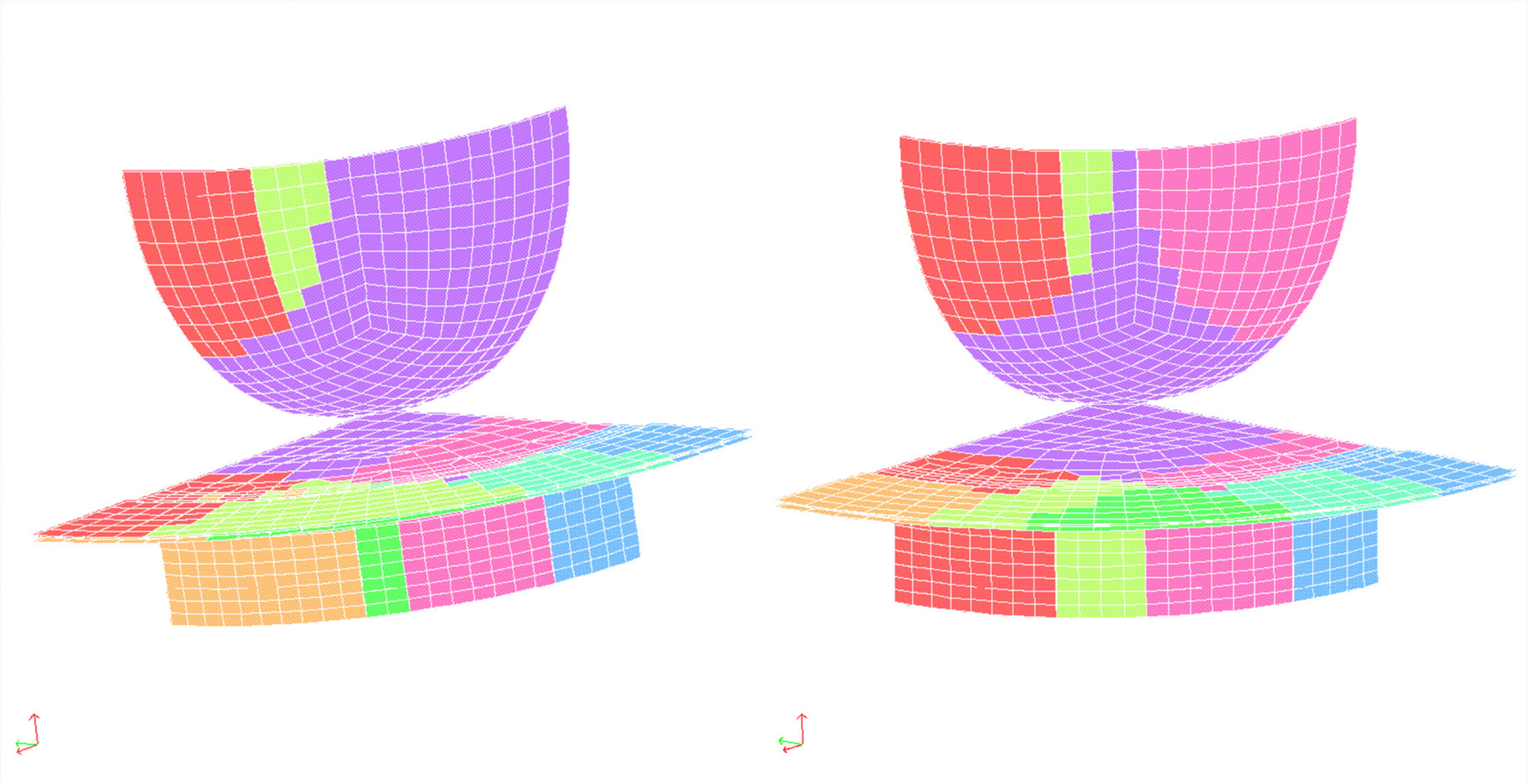
Case Study for Metal Forming: CDD



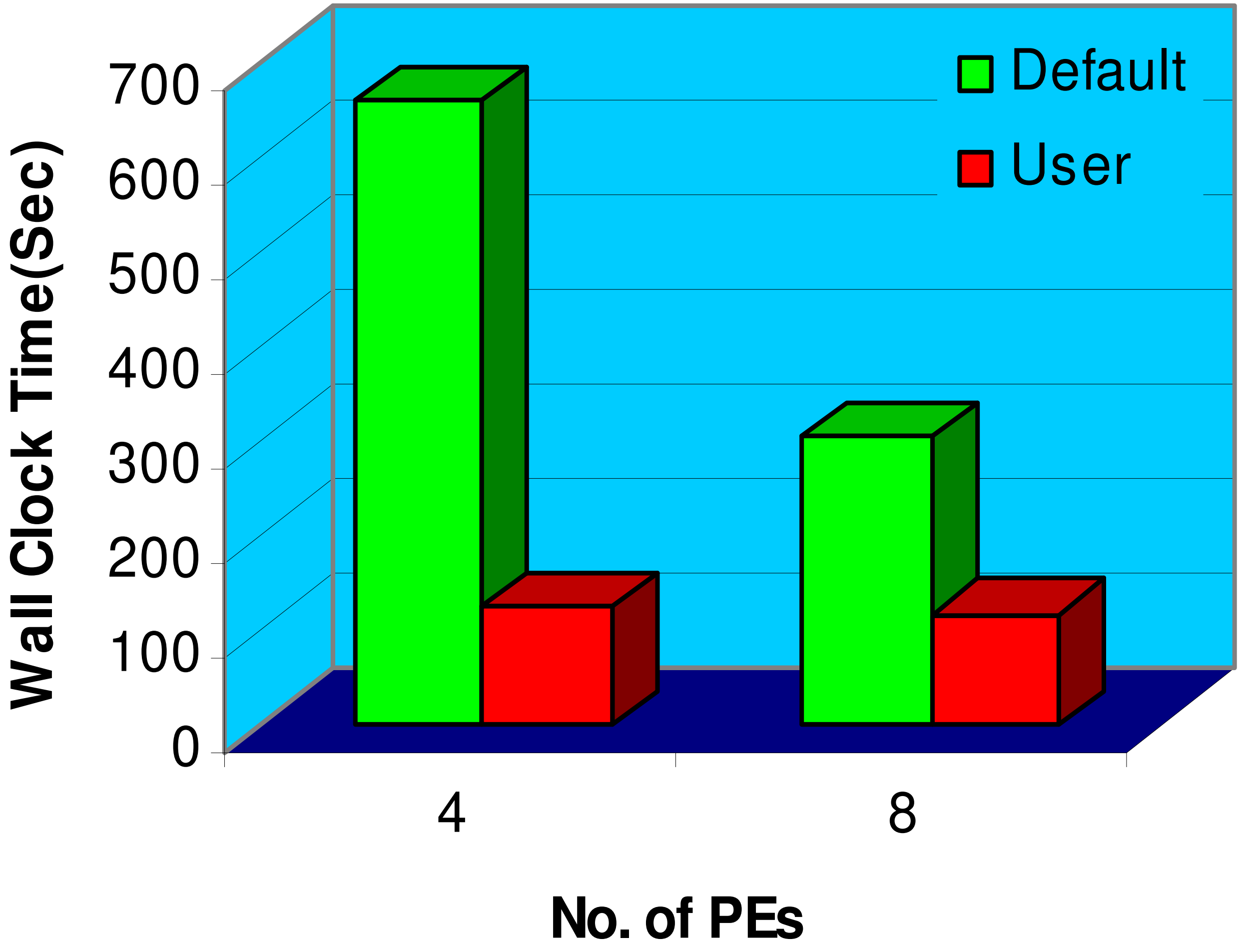
Case Study for Metal Forming: CDD

Default RCB

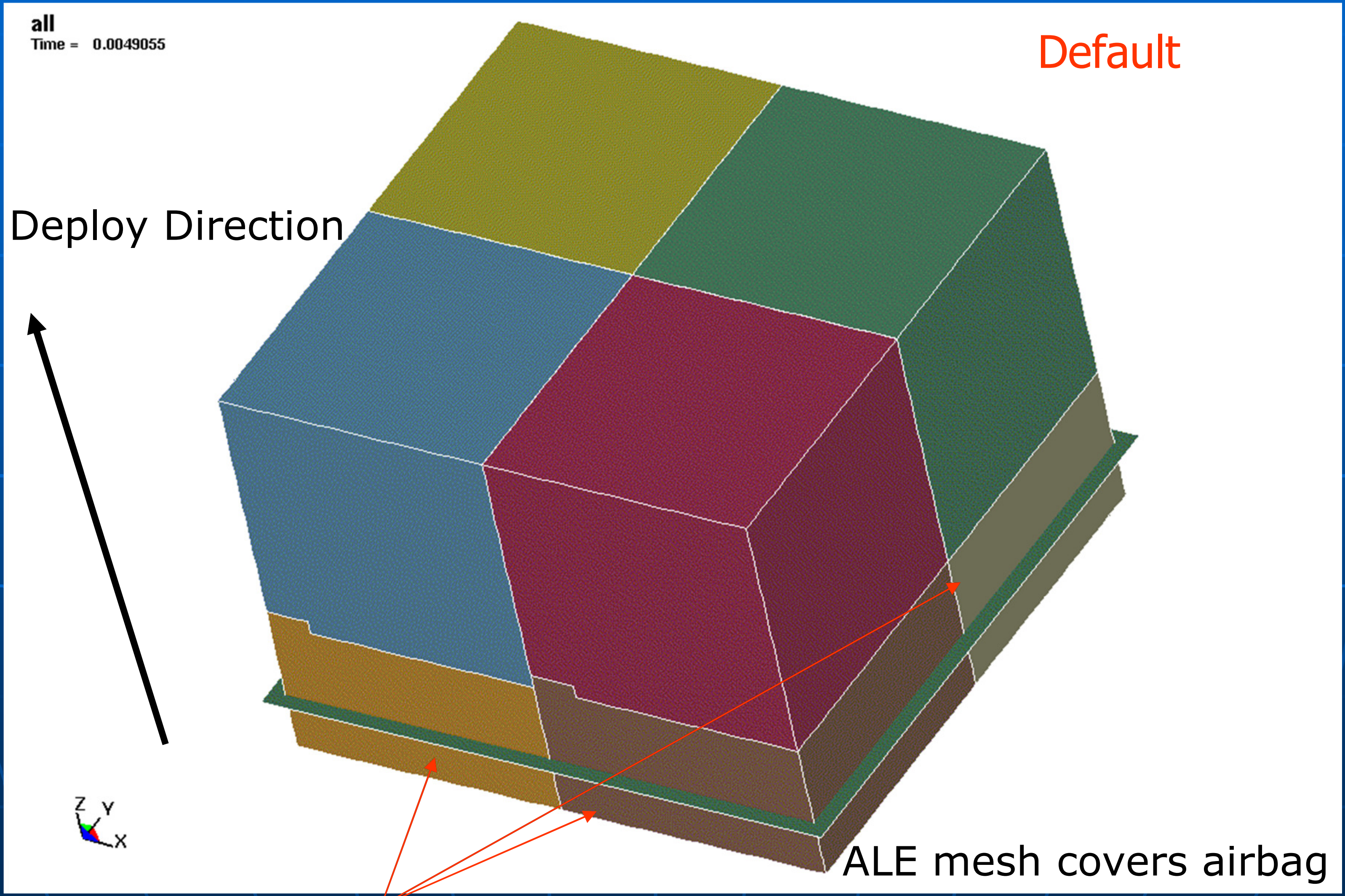
sz 0.



Case Study for Metal Forming: CDD



Case Study for ALE



Default

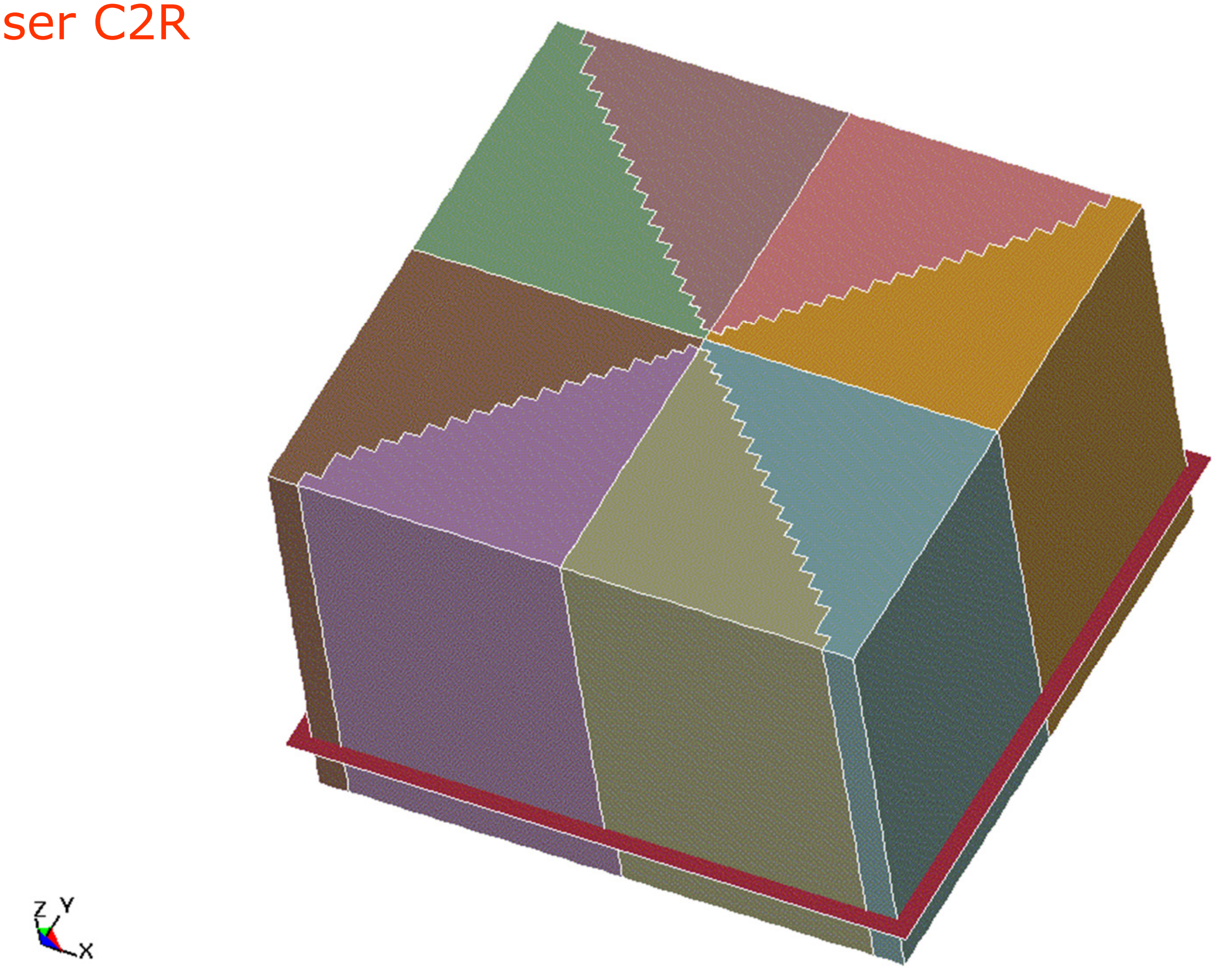
Deploy Direction

ALE mesh covers airbag

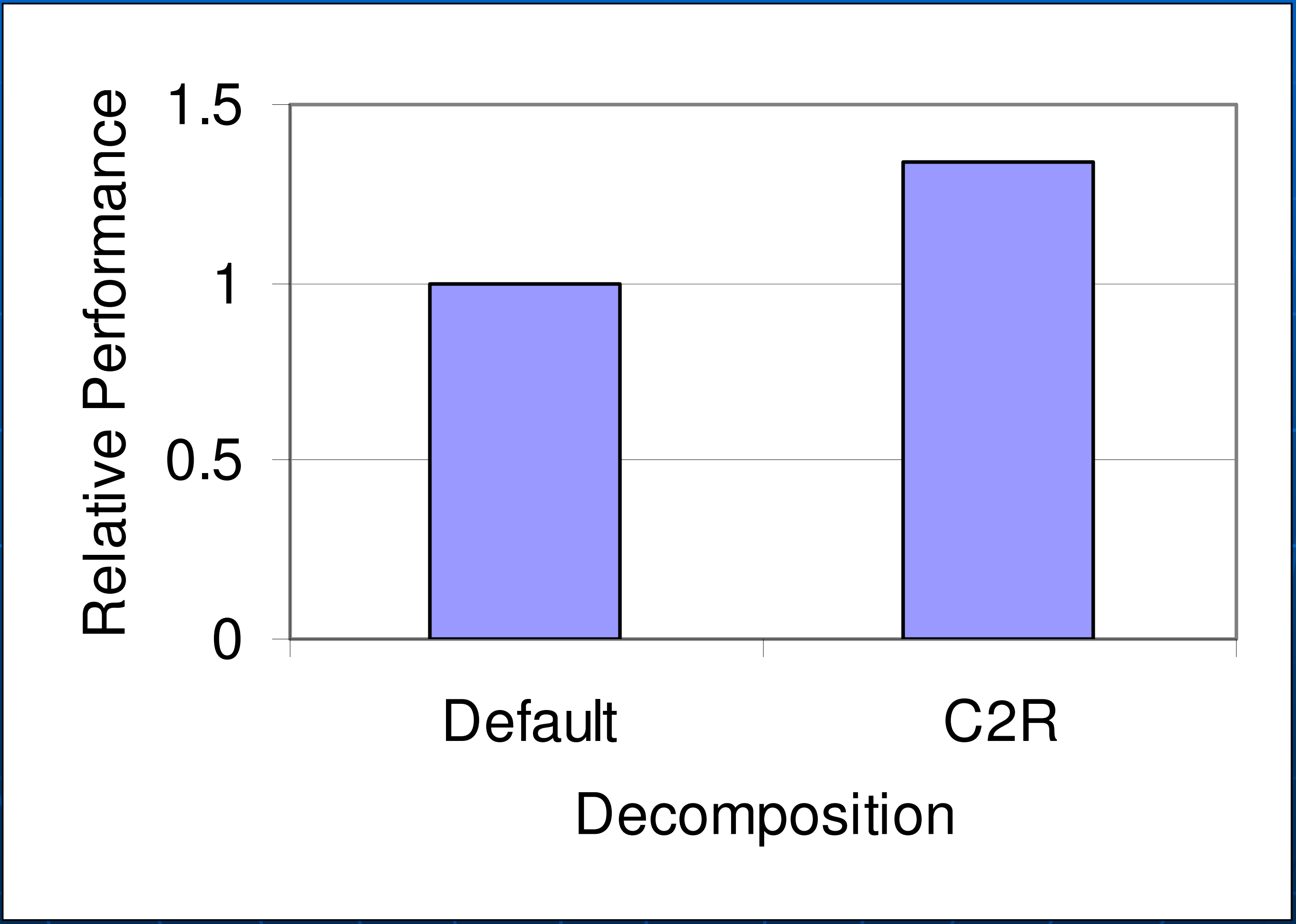
Only 4 CPU’s takes load in the beginning

Case Study for ALE

User C2R



ALE Airbag Timing Comparison first 5000 cycles, 8 CPU



General Guidelines

- Use local file system "dir { local path }" if possible
  This allows MPP job to have scalable IO bandwidth

- Store end results via "dir { rep path }" to the share file system
  The files are moved through MPI calls which has higher bandwidth than NFS file system

- Distribute expansive features or elements to all processors
  i.e. CPM airbag, ALE elements, SPH elements, etc
  (*CONTROL_MPP_DECOMPOSITION_BAGREF)
  (*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_ALE_ELE, etc)

- For number of processors < 16, try to partition model along the direction of initial velocity (use e.g. automatic decomposition (*CONTROL_MPP_DECOMPOSITION_AUTO)

General Guidelines

- Merge small contact definitions into big one

- Distribute large contact area evenly among processors via pfile

  decomp { SILIST 1,2,3 }

Or in input deck

**CONTROL_MPP_DECOMPOSITION_CONTACT_DISTRIBUTE*

- In forming simulation make the decomposition in the direction of the punch travel

- *Please see more pfile options in Appendix O of the user manual The optimal decomposition is model and CPU depended.*

Restart and Pre-decomposition

Restart

- Restart is in MPP-DYNA is different from LS-DYNA, The files are called d3dump##.xxxx or d3fulll##, where ## is a number.

  Simple restart: mpirun –np 5 mpp970 r=d3dump09

  MPP-DYNA finds the child files

  Small restart: mpirun –np 5 mpp970 i=small.k r=d3dump09

  The small restart may have problems. If it does, please report it to LSTC and we will fix it.

  Full restart: mpirun –np 5 mpp970 i=full.k n=d3full09

  Remember *stress_initialization in the inputdeck
  Can change ncpu in full restart
  The full restart can have problems

- Since the Small and Full restart can give problems – check carefully the results

Restart

- Can do stamping in MPP and implicit springback in SMP. Important since implicit is under development in MPP-DYNA 971

- Since the Small and Full restart can give problems – check carefully the results

Pre-decomposition

- Mesh is getting finer and memory requirement increases. Since the decomposition is done on the primary processor, it needs great amount of memory.
- Due to the economy reason, the memory on cluster is limited – 2GB/core.
- It is easier to decompose model in a separated machine with lots of memory.

Run 1: Keyword to structure

```
mpirun –np 1 path_to_mpp/mpp971 i=input.k outdeck=t memory=800m
```

This will convert the keyword input “input.k” to structure file “dyna.str” and stop the execution

Pre-decomposition

Run 2: Create pre-decompose file

pfile:

```
decomp { numproc 16 file input_de }
```

```
mpirun –np 1 path_to_mpp/mpp971 i=dyna.str p=pfile memory=800m
```

This will create pre-decomp database for 16 domains and write necessary information into “input_de.lsda” file. Please note, the job could be restart on a cluster with a node number divided in whole.

Run 3: Restart MPP job on clusters

Move pfile and input_de.lsda to the working directory of target clusters

```
mpirun –np 8 path_to_mpp/mpp971 i=dyna.str p=pfile memory=100m
```

Job could start on clusters with much less memory requirement.

Pre-decomposition

Huge Model > 50M Elements

Run 1: Keyword to structure

```
mpirun –np 1 mpp971_d i=input.k outdeck=t memory=10G
```

Run 2: Create pre-decompose file

```
mpirun –np 1 mpp971_d i=dyna.str p=pfile memory=10G 32ieee=yes
```

Run 3: Restart MPP job on clusters

Move pfile and input_de.lsda to the working directory of target clusters

```
mpirun –np 256 mpp971_s i=dyna.str p=pfile memory=500m 32ieee=yes
```

General Guidelines and Debugging

General Guidelines

- If error termination or unstable behavior occur, check for unsupported features. There is in general no error trap that indicates that a feature not is in MPP.

- 12-32 processors is sometimes preferred for smaller models but the optimal number of CPU's strongly depends on the model.

- Single processor performance of LS-DYNA/MPP ~= LS-DYNA/SMP

- Will run efficiently with large contact definition – ease of modeling

- MPP is beneficial for more than 10k elements/processor

- If contact problems occur
  - Turn on IGNORE option
  - Try to use SOFT=2 at Optional card A.

General Guidelines

consistency

- Same decomposition = same answer

- Changing number of processors < 5% variation in results for well defined model. During the model development, try to keep same number of cores for the analysis. (new Hybrid could be tried to reduce the difference, see the "Recent Development" section).

- Double precision may not help, finer mesh will help for the numerical variations

- Use good engineering judgment to perform special decomposition to reduce numerical variations

General Guidelines

consistency

LSTC_REDUCE

general { lstc_reduce }

Problem: Results changes while changing from dual core to quad core system while using same number of MPP processors

Solution: Fixed summation operation is performed in the code

RCBLOG

decomposition { rcblog filename }

Problem: Decomposition changes during model development

Solution: Preserve the cut line for subsequent runs to reduce the decomposition noise

General Guidelines

Load balancing

- Use local file system "dir { local path }" if possible
  This allows MPP job to have scalable IO bandwidth

- Store end results via "dir { rep path }" to the share file system
  The files are moved through MPI calls which has higher bandwidth than NFS file system

- Distribute expansive features or elements to all processors
  i.e. CPM airbag, ALE elements, SPH elements, etc
  (*CONTROL_MPP_DECOMPOSITION_BAGREF)
  (*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_ALE_ELE, etc)

- For number of processors < 16, try to partition model along the direction of initial velocity (use e.g. automatic decomposition (*CONTROL_MPP_DECOMPOSITION_AUTO)

General Guidelines

Load balancing

- Merge small contact definitions into big one
- Distribute large contact area evenly among processors via pfile

decomp { SILIST 1,2,3 }

Or in input deck

**CONTROL_MPP_DECOMPOSITION_CONTACT_DISTRIBUTE*

- In forming simulation make the decomposition in the direction of the punch travel
- *Please see more pfile options in Appendix O of the user manual The optimal decomposition is model and CPU depended.*

Debugging

- The error messages from MPP-DYNA can be different from LS-DYNA®

- To locate an error one often has to search each of the messag files mes#### in order to find any information. These files are written for each processor.

- The code will trap the segmentation violation (SEGV) and output the rank number. One could rerun the job and attach the debugger to the running thread and get the trace back map. This usually gives good information for changing input.

```
gdb path_to_mpp_code/mpp971 PID
> continue
SEGV
> where
```

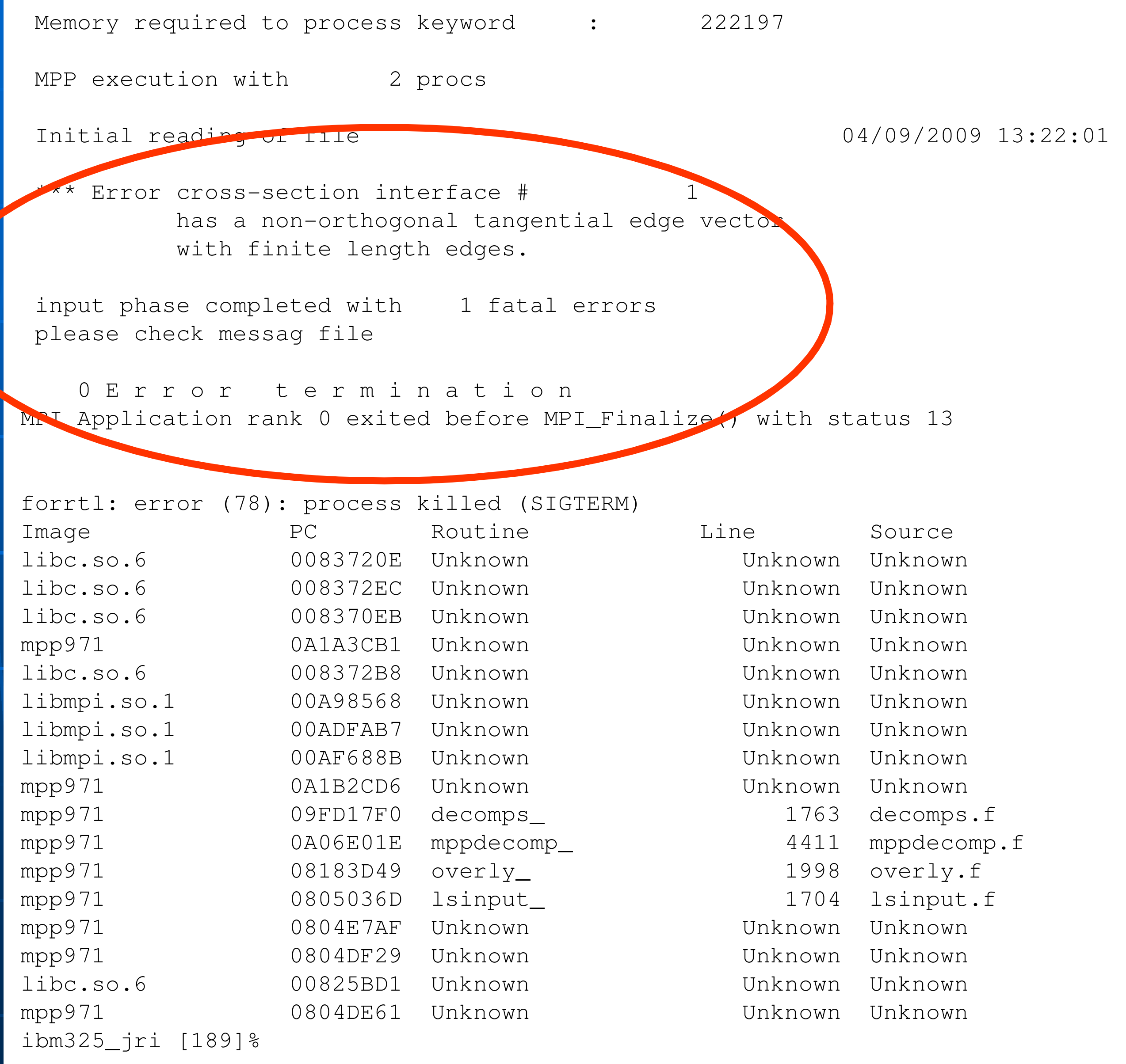
- As for LS-DYNA® a debugger can be used if a core file is written:

```
gdb path_to_mpp_code/mpp971 core
```

  - Type where to get more info and quit for exit
  - Can indicate which subroutine is the problem and hence ease the model debugging.

Debugging

Problem



```
 Memory required to process keyword     :       222197

 MPP execution with       2 procs

 Initial reading of file                                        04/09/2009 13:22:01

 *** Error cross-section interface #           1
          has a non-orthogonal tangential edge vector
          with finite length edges.

 input phase completed with    1 fatal errors
 please check messag file

     0 E r r o r   t e r m i n a t i o n
MPI Application rank 0 exited before MPI_Finalize() with status 13


forrtl: error (78): process killed (SIGTERM)
Image              PC        Routine            Line        Source
libc.so.6          0083720E  Unknown               Unknown  Unknown
libc.so.6          008372EC  Unknown               Unknown  Unknown
libc.so.6          008370EB  Unknown               Unknown  Unknown
mpp971             0A1A3CB1  Unknown               Unknown  Unknown
libc.so.6          008372B8  Unknown               Unknown  Unknown
libmpi.so.1        00A98568  Unknown               Unknown  Unknown
libmpi.so.1        00ADFAB7  Unknown               Unknown  Unknown
libmpi.so.1        00AF688B  Unknown               Unknown  Unknown
mpp971             0A1B2CD6  Unknown               Unknown  Unknown
mpp971             09FD17F0  decomps_                 1763  decomps.f
mpp971             0A06E01E  mppdecomp_               4411  mppdecomp.f
mpp971             08183D49  overly_                  1998  overly.f
mpp971             0805036D  lsinput_                 1704  lsinput.f
mpp971             0804E7AF  Unknown               Unknown  Unknown
mpp971             0804DF29  Unknown               Unknown  Unknown
libc.so.6          00825BD1  Unknown               Unknown  Unknown
mpp971             0804DE61  Unknown               Unknown  Unknown
ibm325_jri [189]%
```

In MPP the error can look serious!

Debugging

```
WRITE ERROR: iam=0  file=d3plot  which=34  where=8192 wrote 0 of 65536
   52562 t 1.7000E-03 dt 3.17E-08 write d3plot file
```

This means that there is no disk space on node 0 (the iam tells the nodenumber). Notice that on some machines the "no space left on device" message will not be showed, this is the case for Linux Cluster.

Debugging

This error was from a MPP Linux run:

```
Performing Recursive Coordinate Bisection

p1_3586: (479.788216) xx_shmalloc: returning NULL; requested 1585896 bytes
p1_3586: (479.788313) p4_shmalloc returning NULL; request = 1585896 bytes
You can increase the amount of memory by setting the environment variable
P4_GLOBMEMSIZE (in bytes)
p1_3586:  p4_error: alloc_p4_msg failed: 0
bm_list_3583:  p4_error: net_recv read:  probable EOF on socket: 1
```

p4 error is normal from MPICH, i.e. this is a MPI error, in this case is suggested to set an environment variable

Debugging

```
*** Error Memory is set        1235165 words short
        Current memory size        50000000
        Increase the memory size by one of the following
        where #### is the number of words requested:
         1) On the command line set - memory=####
         2) In the input file define memory with *KEYWORD
           i.e., *KEYWORD #### or *KEYWORD memory=####
```

- The memory unit is in WORD. For single precision is 4 Bytes/word and for double precision is 8 Bytes/word.

- LS-DYNA® explicit uses real memory to store all data. However, the amount of static memory requested is controlled by “memory=“ option and the amount of dynamic memory is adjusted automatically.

- Please use “top” command to check the available memory in the system and you ***DO NOT*** want your job using swap space

Recent Development

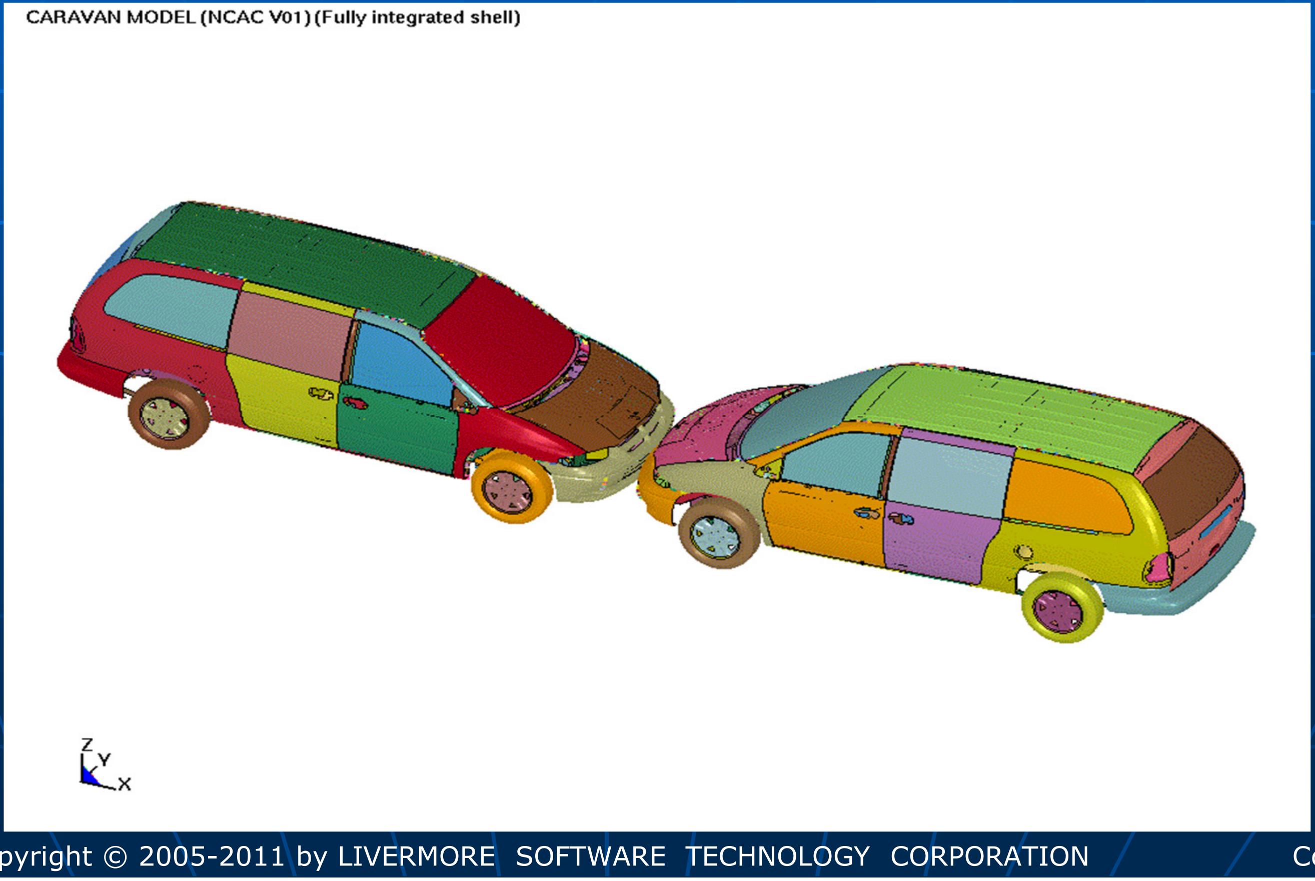
Current Development

- Many new options are implemented in MPP-DYNA in recent years. Both in versions of 970 and 971
  - Pinball Contact (SOFT = 2) - 970
  - ALE FSI applications - 970
  - SPH method – 970
  - Automatic decomposition - 970
  - Implicit solvers - 971
  - EFG – 971
  - Thermal – 971
  - Particle Method 971
  - ....

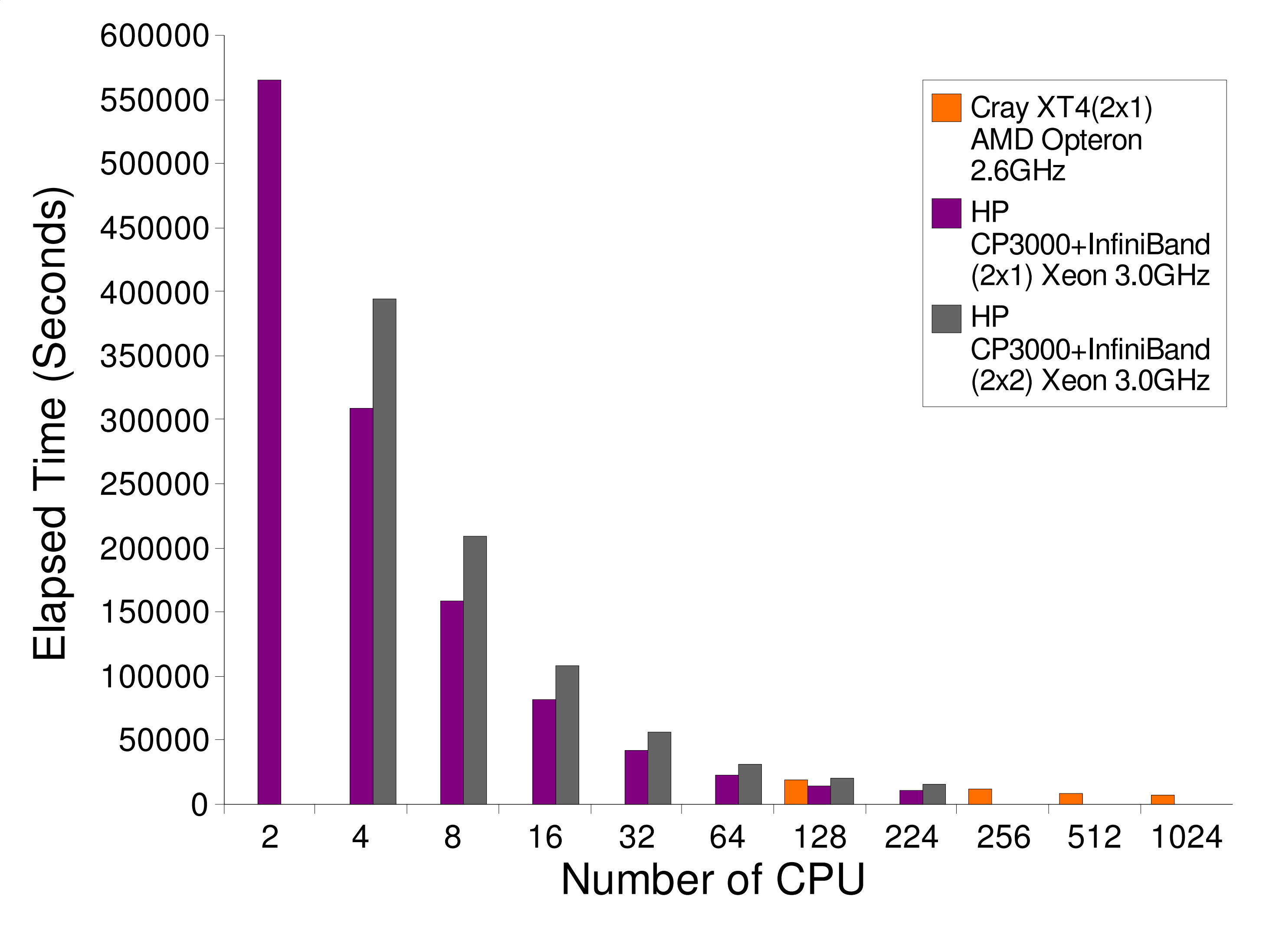
LSTC Livermore Software Technology Corp.

Scalability on Large Number of CPUs

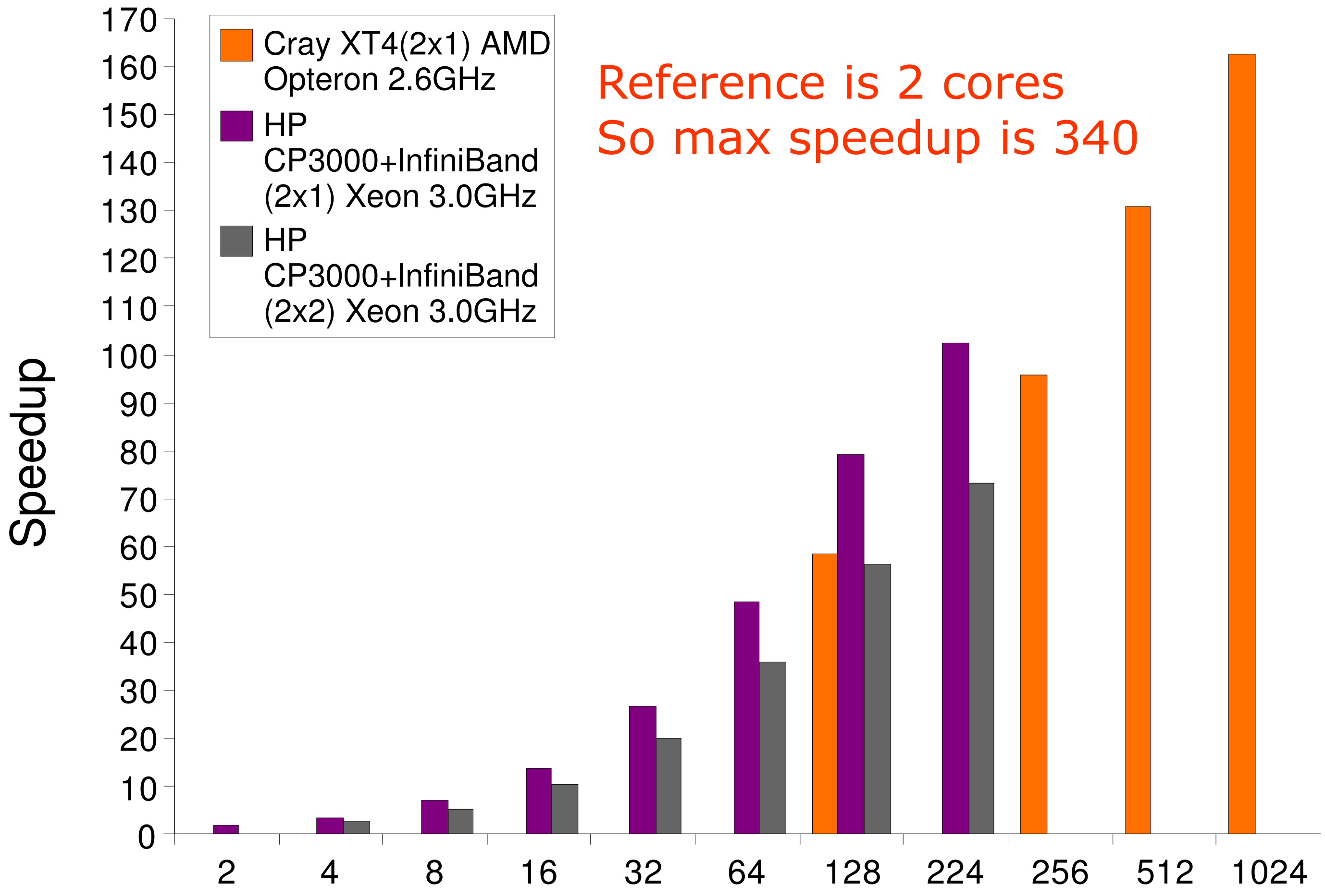
Model statistic (car2car model)
~2,500,000 nodes and elements
53 contacts
Fully integrated (type 16) shells



Scalability on Large Number of CPUs



Scalability on Large Number of CPUs



Reference is 2 cores
So max speedup is 340

Note: Not ideal scaling for large number of CPU's

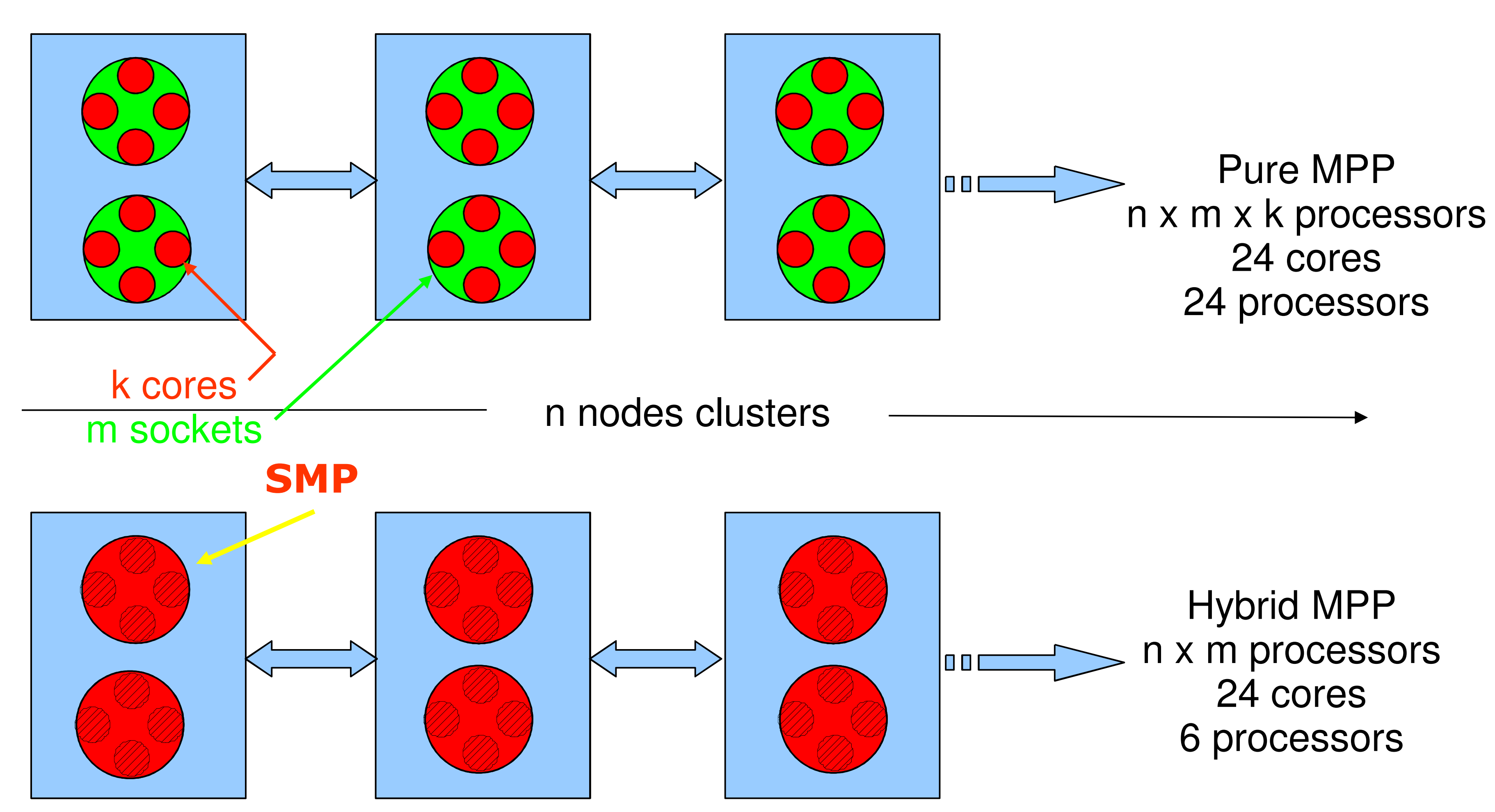
Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters

- It has been seen that scaling for a large number of processors, typically larger than 128, not always is good.
- Sometimes the results can varies with number of CPU's due to the decomposition, especially if the model is unstable.
- A new approach is currently being tested, it runs SMP within each CPU and MPP between the CPU's.
- It is named Hybrid.
- If the number of SMP threads is increased it will give identical results.
- To run Hybrid both SMP and MPP variables will have to be set.

Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters



k cores

m sockets

n nodes clusters

SMP

Pure MPP
n x m x k processors
24 cores
24 processors

Hybrid MPP
n x m processors
24 cores
6 processors

LSTC
Livermore Software
Technology Corp.

Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters

- There is a special syntax that is required for the Hybrid approach.

- If e.g. the set-up is a system with 16 nodes, dual socket quad core system (as previous slide) the variable is:
  - Set OMP_NUM_THREAD=4 (max four cores in each SMP)
  - The system is a 128 core system

- mpirun –np 32 mpp971_hybrid i=input ncpu=-1
  - 32 MPP Processors (green circle) and 1 core in each which then is a total of 32 cores.

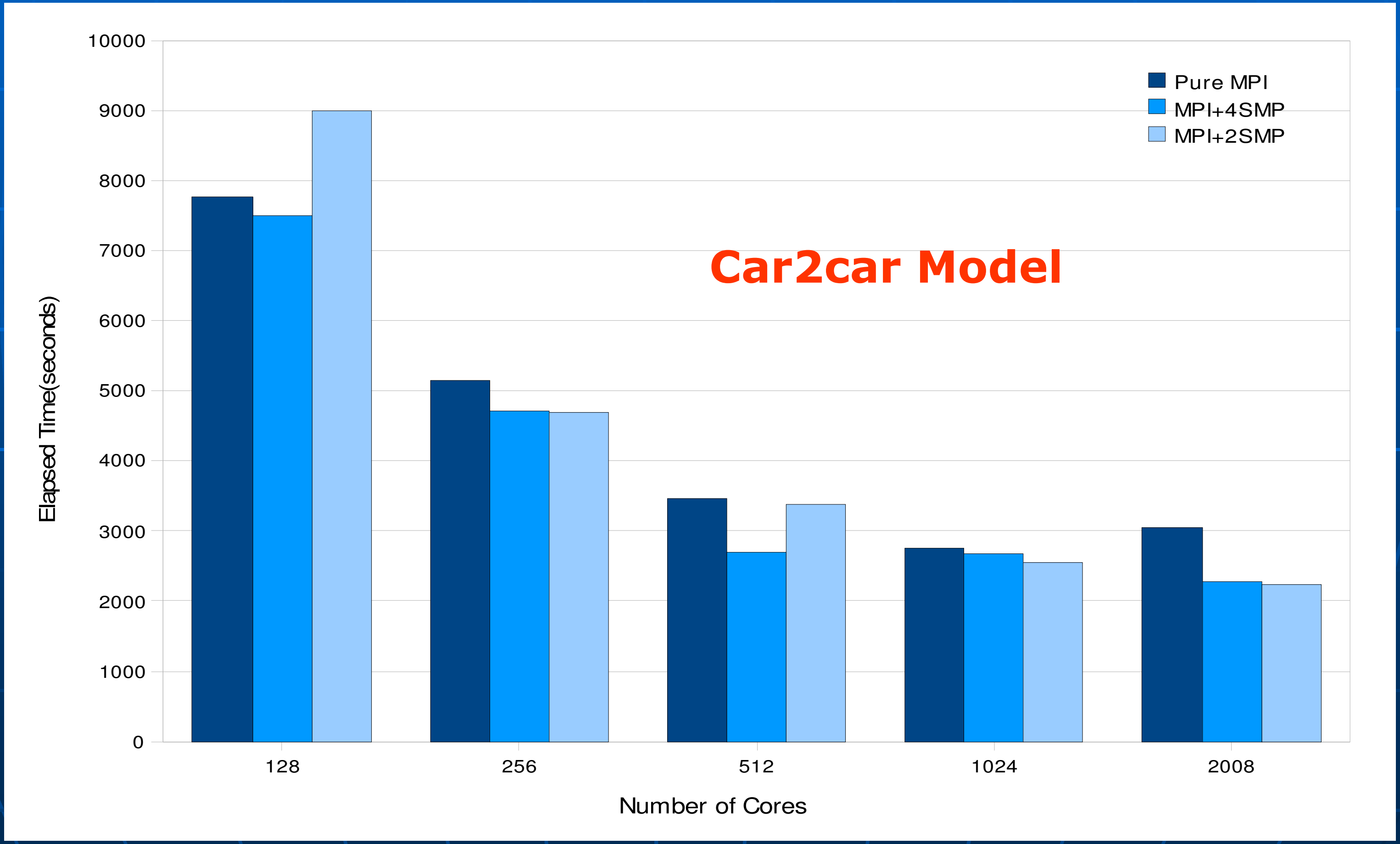
- mpirun –np 32 mpp971_hybrid i=input ncpu=-2
  - 32 Processors and 2 cores in each = 64 cores

- mpirun –np 32 mpp971_hybrid i=input ncpu=-4
  - Total of 128 cores is used

Explicit MPP/Hybrid Performance

Multi-core/Multi-socket clusters

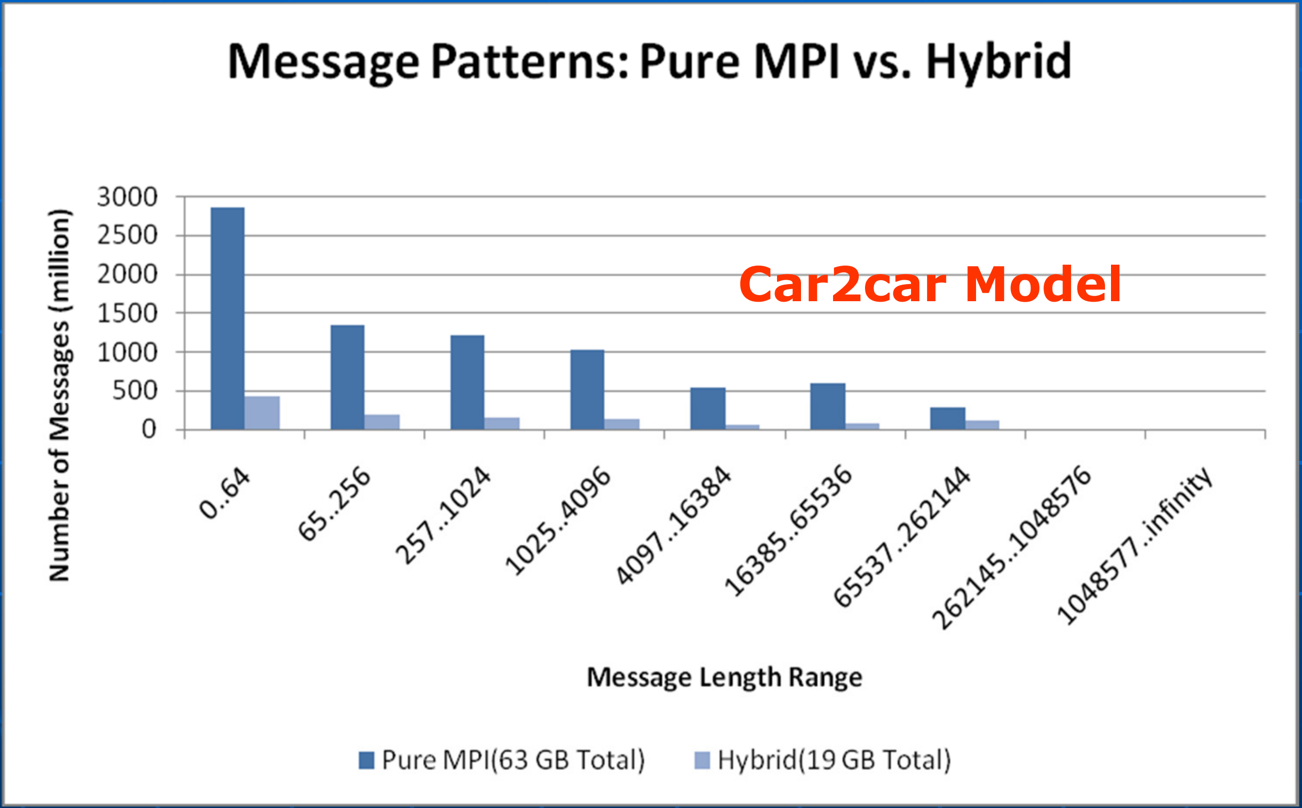
Performance Comparison on Windows Server 2008



Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters

Message Across Network

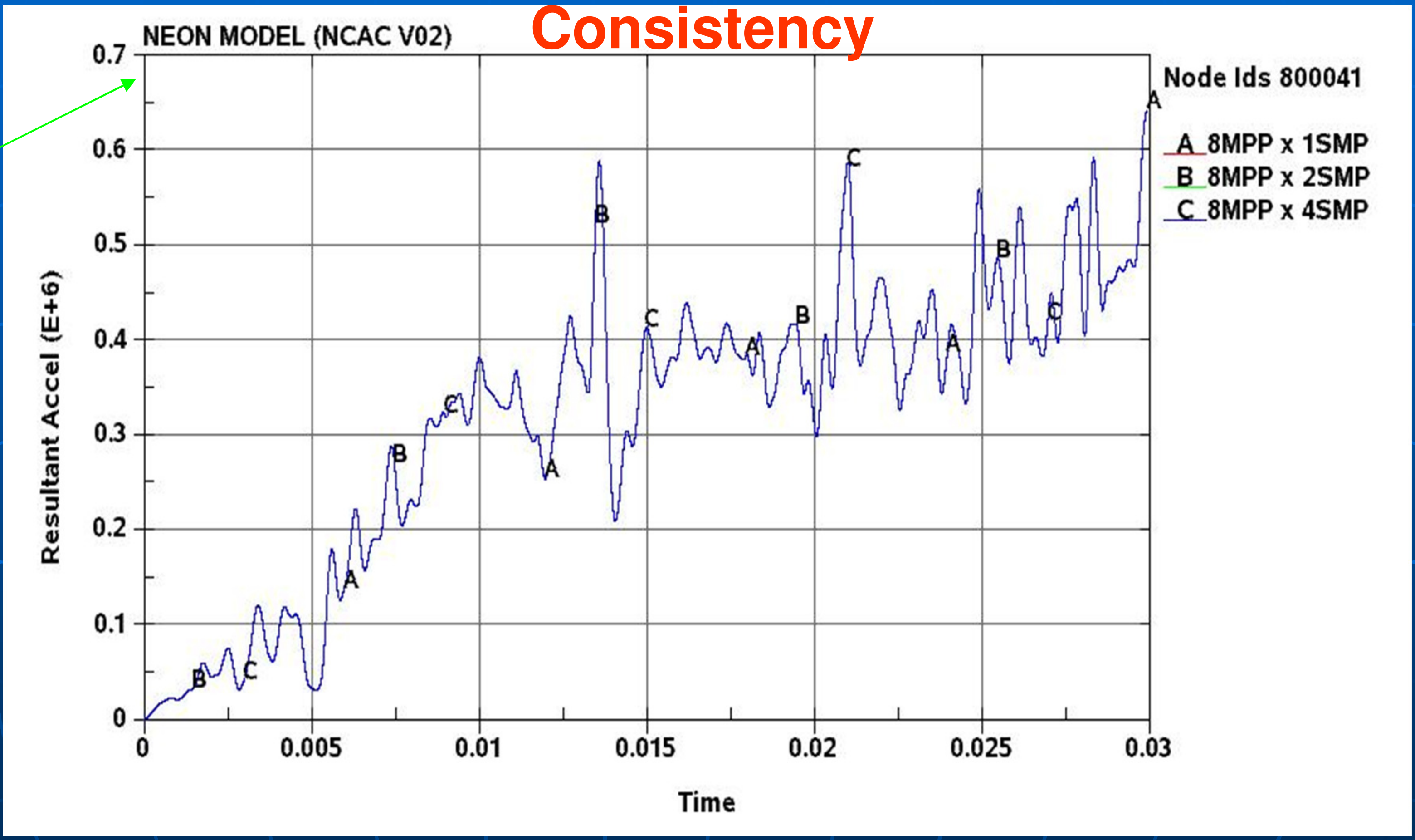


- Hybrid greatly reduce the amount of data through network and provide better scaling to large number of processors

Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters

Note



- Consistent results is be obtained with fix decomposition and changing number of SMP threads

Scalability on Large Number of CPUs

Multi-core/Multi-socket clusters

consistency tests and performance comparison of HYBRID and pure MPP code.

	12p	12x-1	12x-2	12x-4
Case 1	108118	124035	81380	60215
Case 2	75028	85367	50467	33728
Case 3	68047	87924	55599	35773
Case 4	16610	22677	13073	8759
Case 5	36522	44622	28397	20215
Case 6	14253	18898	12169	8705
Case 7	9485	12753	7600	5800
Case 8	937	1260	773	569
Case 9	12640	16012	10486	6926

Implicit MPP/Hybrid Performance

Multi-core/Multi-socket clusters

Performance on Linux AMD64 systems

No. of cores (node x socket x core)	WCT of Factor Matrix (seconds)	WCT for job to complete (seconds)
16 x 4 x 1	2055	14417
16 x 4 x 2	985	13290
16 x 4 x 4	582	29135
16 x 4 x 4omp (Hybrid)	960	9887

Get the best performance from MPP Hybrid

1) Turn hyperthreading off
2) OMP_NUM_THREADS to SMP upper limit
3) General variables for MPI
   - Platform (HP) MPI

     -cpu_bind_mt=MASK_CPU:*string*

     -e MPI_THREAD_AFFINITY=packed
   - Intel MPI

     -env I_MPI_PIN_DOMAIN=*string*

     -env I_MPI_PIN_ORDER=compact

     -env KMP_AFFINITY=compact

Get the best performance from MPP Hybrid

- How to find out the *string*
- Find the core ordering

  cat /proc/cpuinfo | grep –i “physical id”

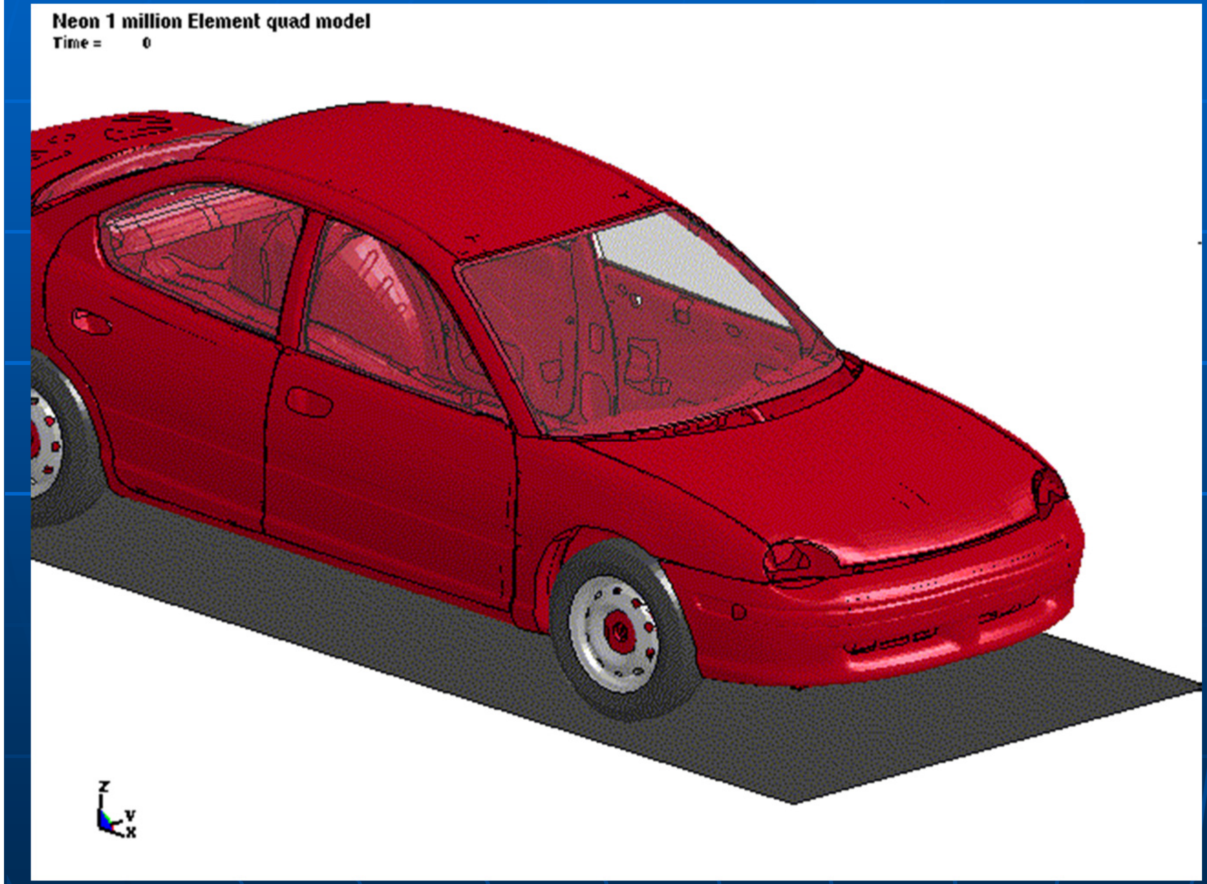
  Example: Dual 6 cores 0 0 0 0 0 0 1 1 1 1 1 1
- Pin application to cores sharing local resource

  Example: 3 SMP/MPP on each node

	CPU #1						CPU #0						HEX #
1st MPP	0	0	0	0	0	0	0	0	0	1	1	1	7
2nd MPP	0	0	0	0	0	0	1	1	1	0	0	0	38
3rd MPP	0	0	0	1	1	1	0	0	0	0	0	0	1C0
4th MPP	1	1	1	0	0	0	0	0	0	0	0	0	E00

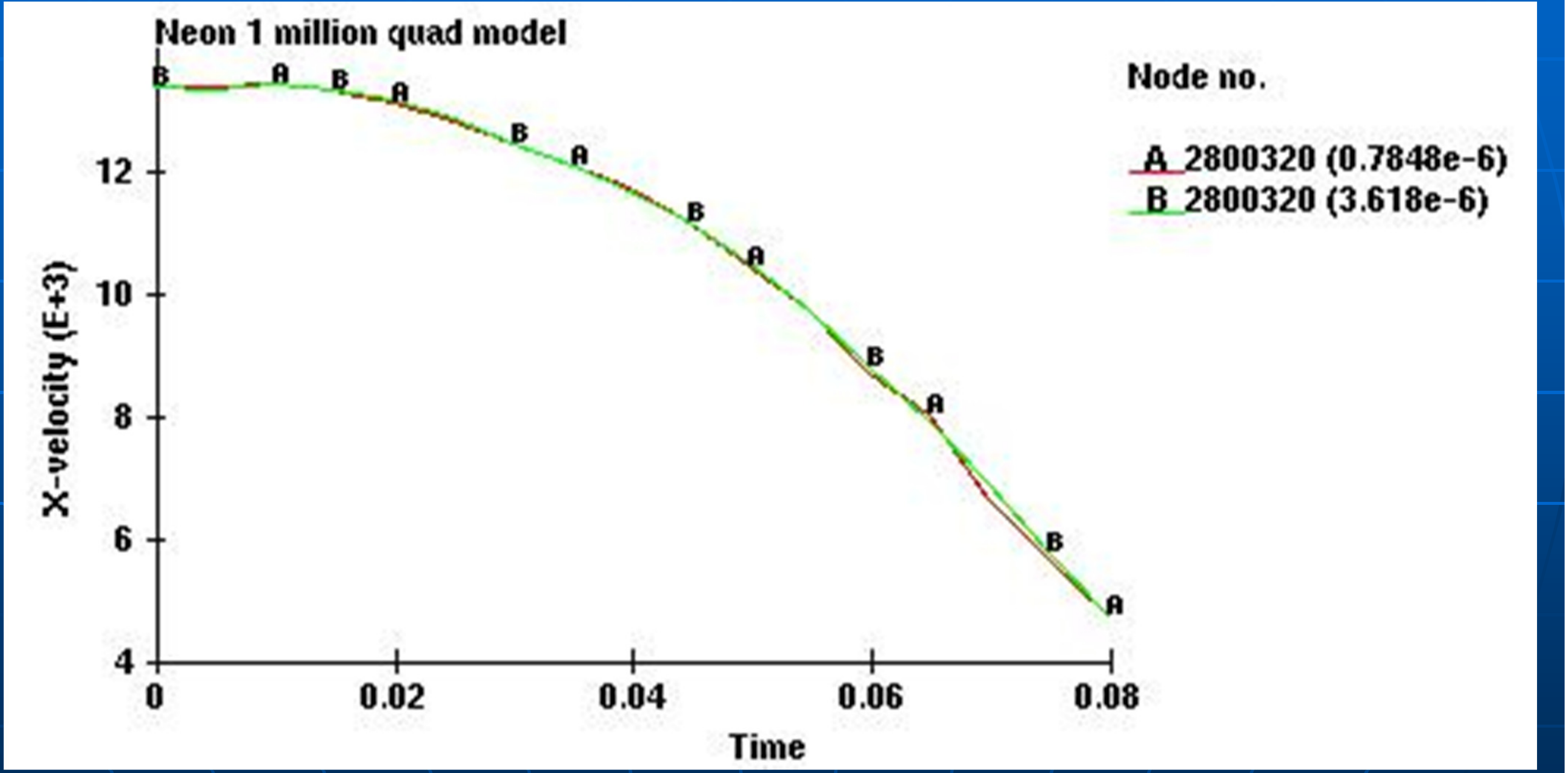
String = *7,38,1C0,E00*

Neon 1 million elements



1056383 quad shells
130 beams
2852 solids
1 contact for the entire model
Termination time 0.080 secs
Timestep 3.618e-6 secs
Ascii and binary outputs disabled.
Pre-decomposed with 1cpu

Neon 1 million elements



130

Neon 1 million elements

128x2x4 dt=7.85e-7 8% mass increase Conventional mass scaling	6 minutes 18 seconds
128x2x4 dt=3.618e-6 894% mass increase Selective mass scaling Ongoing development to support more features for selective mass scaling	5 minutes