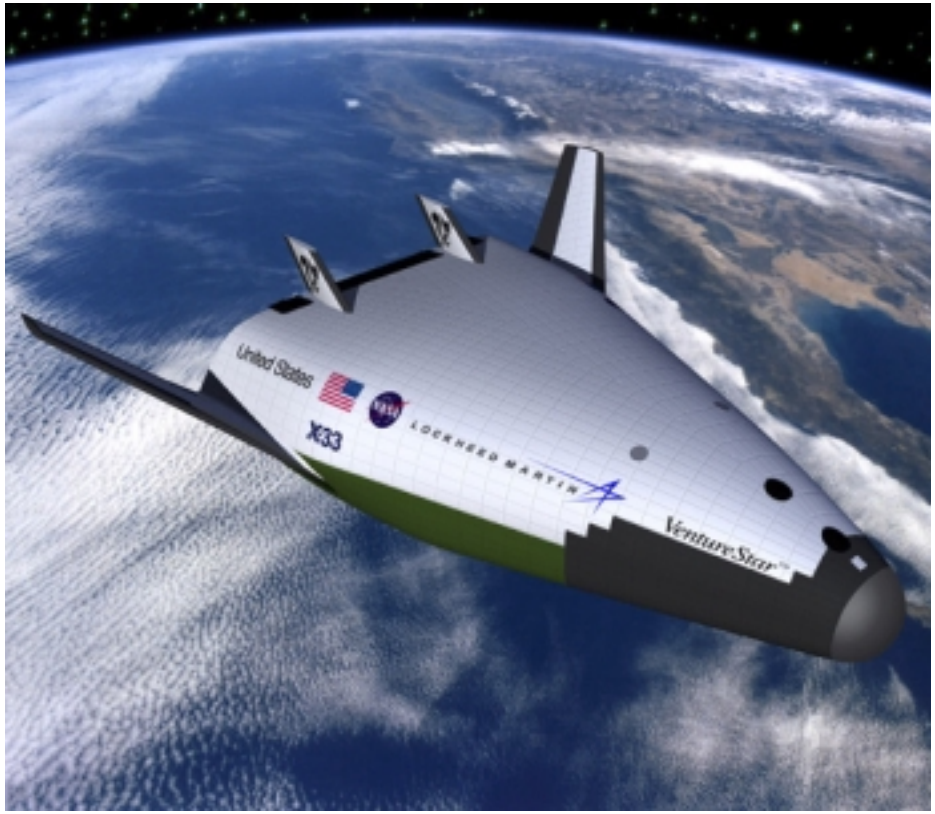


Early Experiences with THE 512p Origin2000

Ciotti, Taft - NAS
Petersohn - SGI

Bob Ciotti
Tera-Scale Application Group Lead
Numerical Aerospace Simulation
NASA Ames Research Center
Moffett Field, CA 94035-1000
Ciotti@nas.nasa.gov

Numerical Aerospace Simulation

- **Tera-Scale Application Group - What do we do?**
  - **Capability Applications**
  - **What "Can't" be done today**

  - **Aeronautics - Traditional Role in CFD**
    - **Overflow - Low Mach**
    - **LAURA - High Mach/Chemistry**
    - **TLNS3D - Low Mach (used in commercial aircraft analysis/design)**
    - **INS3D - Turbulent Incompressible (Turbo-Pump Design)**
  - **Astrobiology - New Institute at Ames Research Center**
    - **Molecular Dynamics**
    - **Stellar Dynamics as it relates to the origin of life**
  - **Weather Modeling**
    - **GCM**
  - **Nano Technology**

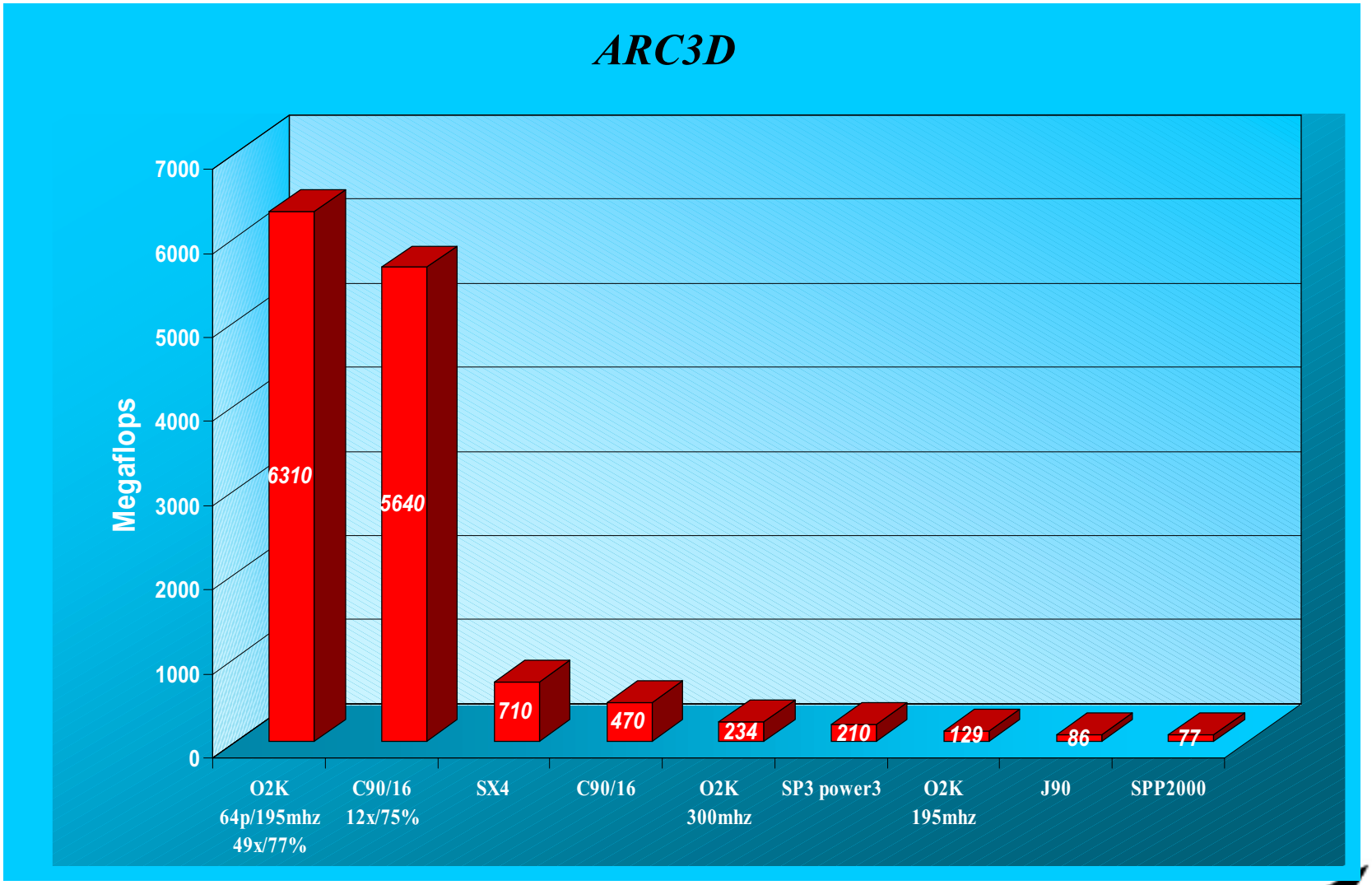
Points of the Talk

- **IRIX Operating System Modifications**

- **Application Performance and Programming Techniques**
  - **Overflow**
  - **Multi-Level Parallelism (MLP) Library**

- **Reliability Analysis**
  - **C90/16-1024 VonNeumann**
  - **256p Origin2000 Steger**
  - **512p Origin2000 Lomax**

Performance



Real Application Performance?

Effort started with concerns about RISC system performance

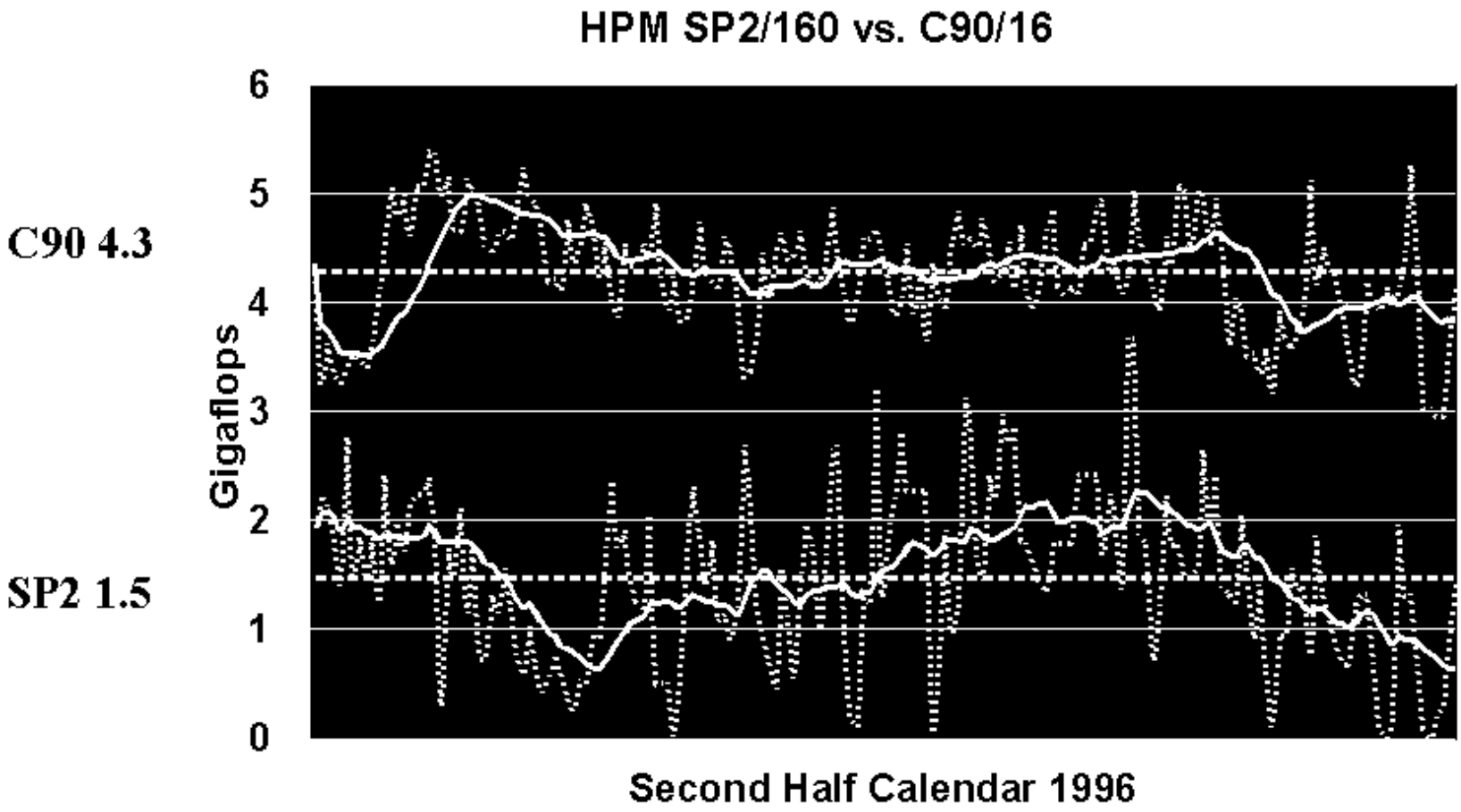
"I am quite concerned that benchmarks for stripped down elements of a CFD code are being confused with performance numbers for a full CFD code, which are usually 2-10 times slower"

"Taking these two things into account, my bet is that we will not beat a C-90 (e.g. 4-6 GFLOP/s) on any SNX architecture. I would love to be proved wrong, but there have been the CM2s, the Hypercubes, the Paragons, the DaVincis, the SP1s, the SP2s, etc. all of which were supposed to outperform the Crays on real problems, and which really never got close."

"Please convince me otherwise"

Crusty Old CFDer (May, 1997)

SP2 vs. C90



+ Facilitated Parallel Applications Development

- Poor Sustained Individual App/Workload Performance

RLV - One Really Hard Problem

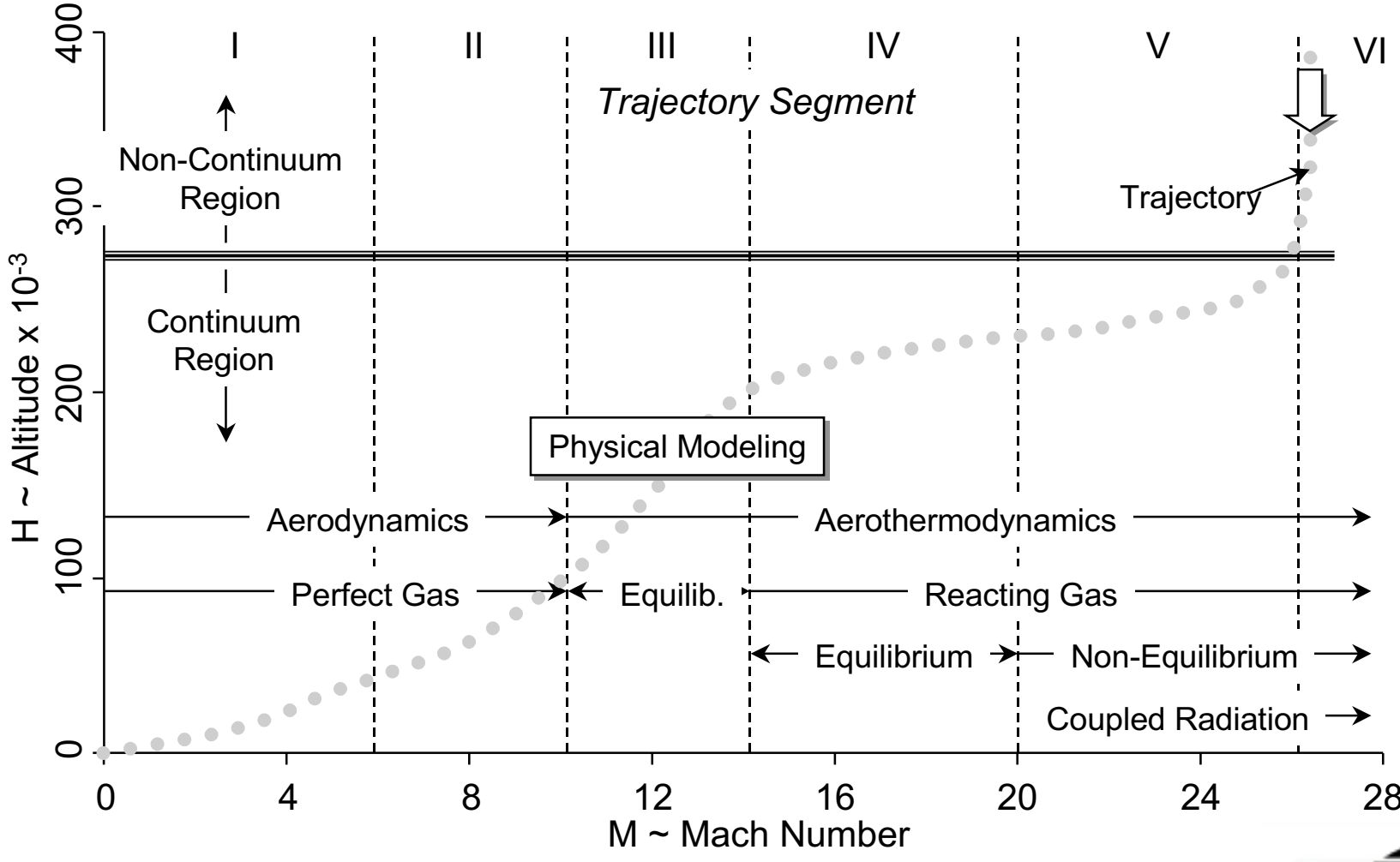
Reusable Launch Vehicle (RLV - what is it?)

- Generic Term for the next generation shuttle replacement
- First "man-rated" vehicle designed since the Shuttle
- First "man-rated" vehicle to be designed on supercomputers

(Shuttle designed without CFD)

(Fact – First Shuttle Launch
April 12, 1981, 7:00:03 a.m. EST)

Typical RLV Descent Trajectory: Aerodynamics Analyses



Estimated RLV Configuration Analysis
(5.6 C90/16 years and 12.6 terabytes storage/configuration)

NASA HEC Requirements Study
Code R Programs
RLV: Aerodatabook Requirements (no TSP/Structures/Materials)

Trajectory Segment		Memory, Mass Storage, & CPU Requirements																		
Segment Identifier	Mach Range	per Code Run							per RLV Trajectory									per RLV Configuration		
		Memory					Mass Storage (GBytes)	CPU Time (C-90 Hrs)	Code Run Multipliers						Mass Storage (GBytes)	CPU Time (C-90 Hrs)	No. of Traj's	Mass Storage (GBytes)	CPU Time (C-90 Hrs)
		Grid Size (MPts)	Variables/Grid Point: Basic	Addt'l	Total	Total (Mbytes)			Mach No.'s	α's	β's	Body Flap δ's	Elevon δ's	Total					
I	0<M<6	8	8	0	8	1,280	3.2	150	5	3	3	3	3	405	1,296	60,750	3	3,888	182,250
II	6<M<10	8	8	0	8	1,280	3.2	150	1	3	3	3	3	81	259	12,150	3	778	36,450
III	10<M<14	8	8	0	8	1,280	3.2	225	1	3	1	3	3	27	86	6,075	3	259	18,225
IV	14<M<20	8	8	11	19	3,040	7.6	300	5	3	1	3	3	135	1,026	40,500	3	3,078	121,500
V	20<M<26	8	8	13	21	3,360	8.4	600	4	3	1	3	3	108	907	64,800	3	2,722	194,400
VI*	M>26	8	8	22	30	4,800	12.0	1,200	2	3	1	3	3	54	648	64,800	3	1,944	194,400

		Totals			
		per Trajectory		per Configuration	
Trajectory Segments I -- V	Mass Storage & CPU Time (Gbytes & C-90 Hrs)	3,575	184,275	10,724	552,825
	CPU Time (Equiv. von Neumann Computational Years**)		1.38		4.15
Trajectory Segments I -- VI	Mass Storage & CPU Time (Gbytes & C-90 Hrs)	4,223	249,075	**12,668**	747,225
	CPU Time (Equiv. von Neumann Computational Years**)		1.87		**5.61**

Trajectory Seg. ID	Physical Model Assumptions
I	Standard Aerodynamics w/Perfect Gas
II	Standard Aerodynamics w/Perfect Gas
III	Aerothermodynamics w/Equilibrium Chemistry
IV	Aerothermodynamics w/Reacting Gas/Equilibrium
V	Aerothermodynamics w/Reacting Gas/Non-Equilibrium
VI	Aerothermodynamics w/Reacting Gas/Non-Equilibrium w/Coupled Radiation

Notes
* Estimated
** 95% utilization assumed

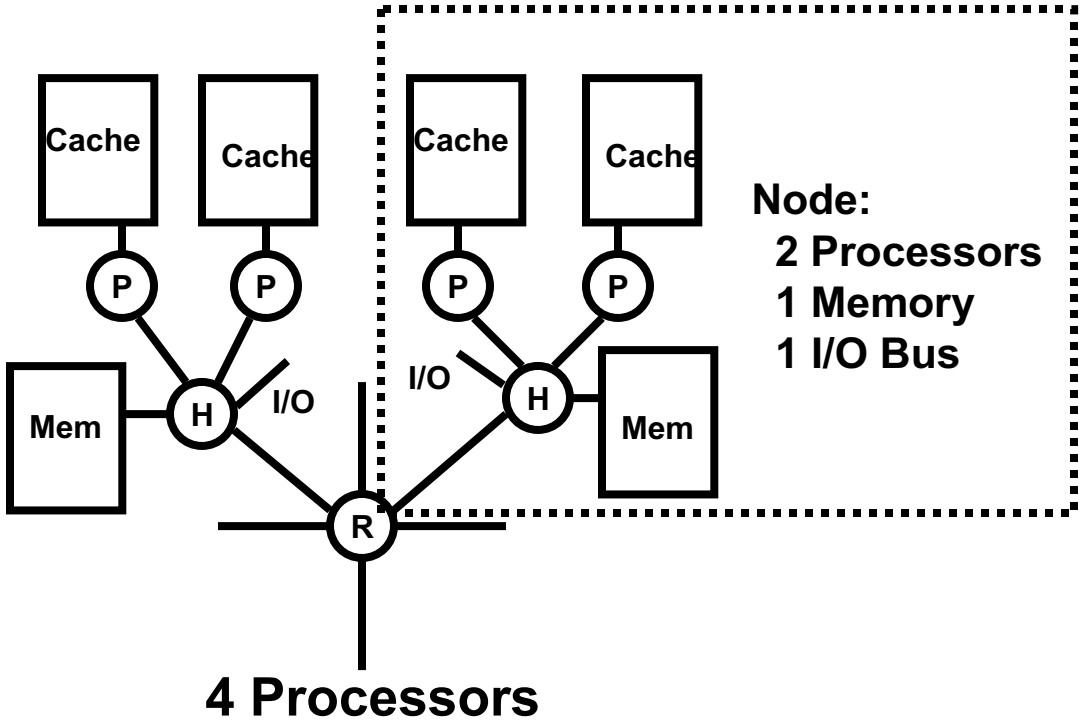
Estimates are Conservative
Minimum # Configurations to Analyze: 3 (16.8 C90/16 Years)

One Configuration - Where Do We Need To Be?

C90/16 Years	C90/16 Perf	Performance in Sustained Gigaflops	Years to Goal at 2x18months	Computational Time (Days)
5.6	4	100		80.64
5.6	4	200	1.50	40.32
5.6	4	400	3.00	20.16
5.6	**4**	**800**	**4.50**	**10.08**
5.6	4	1,600	6.00	5.04
5.6	4	3,200	7.50	2.52
5.6	4	6,400	9.00	1.26
5.6	4	696,500,000	44 Years	1 Second

Architecture

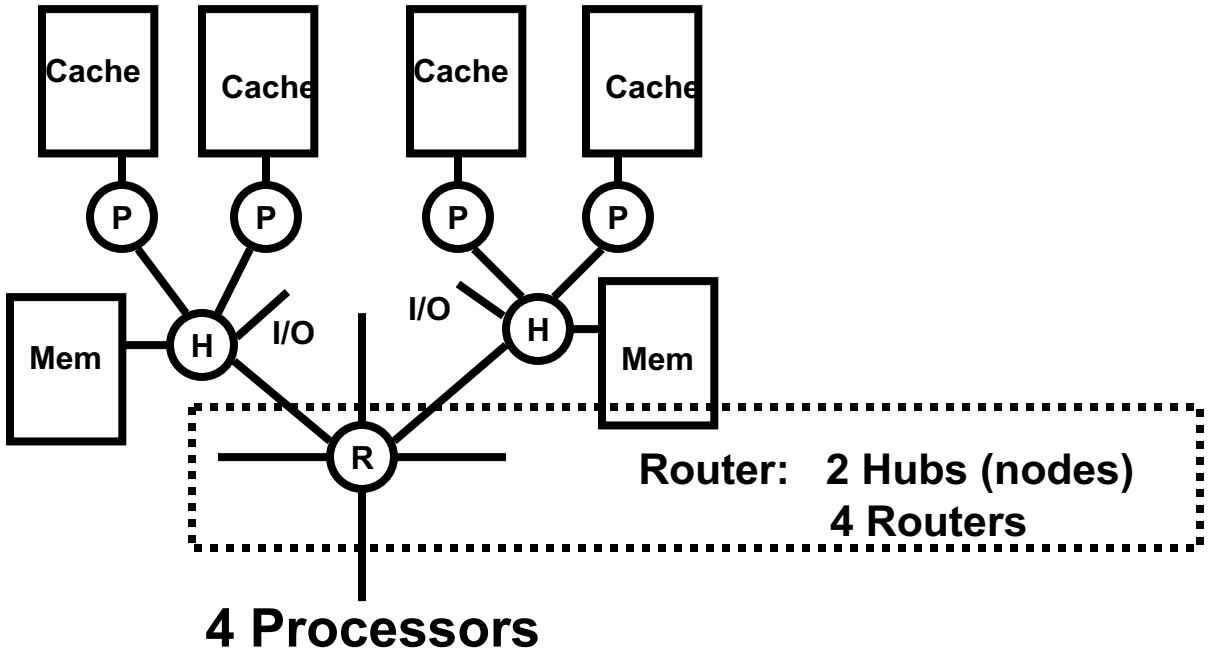
Basic building block of Origin2000 - No other hardware components



Commercially Viable - No unique hardware required for largest systems

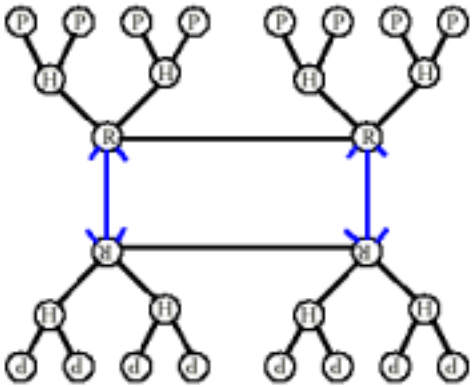
Architecture

Basic building block of Origin2000 - No other hardware components

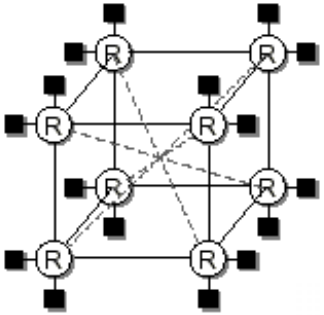


Commercially Viable - No unique hardware required for largest systems

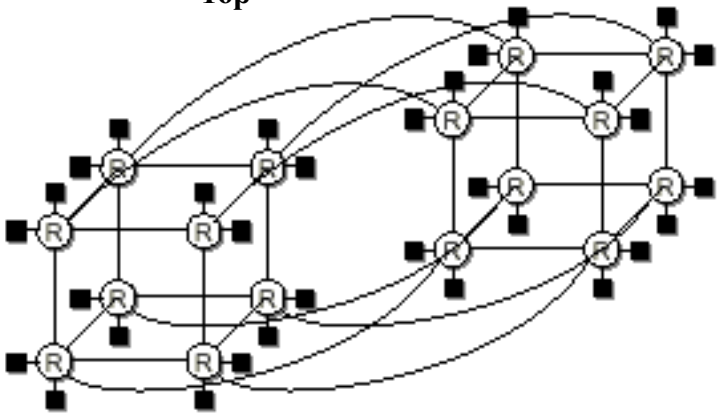
Topology



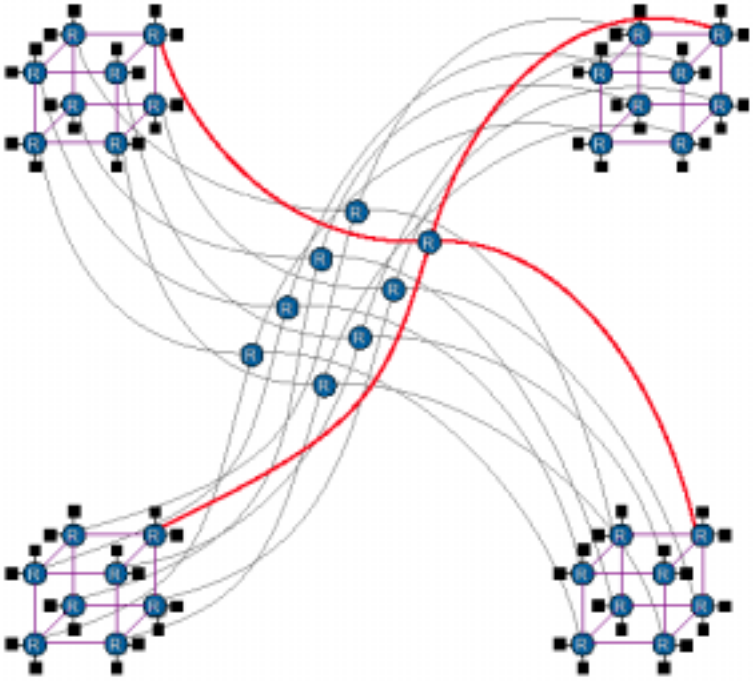
16p



32p

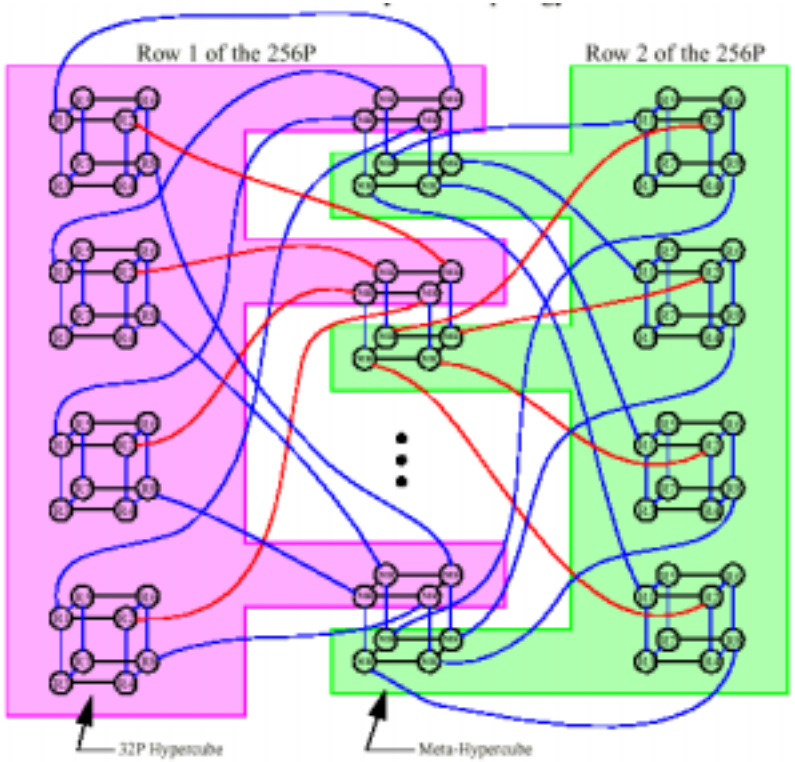


64p

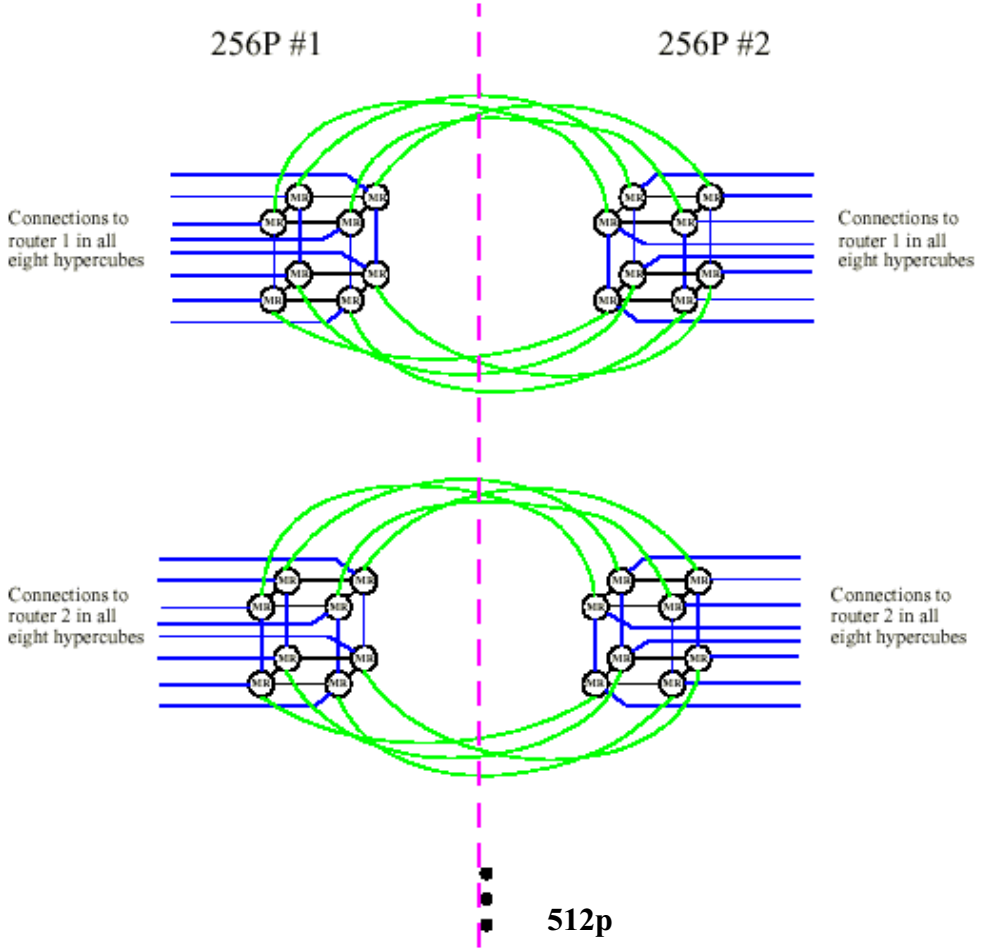


128p

Topology

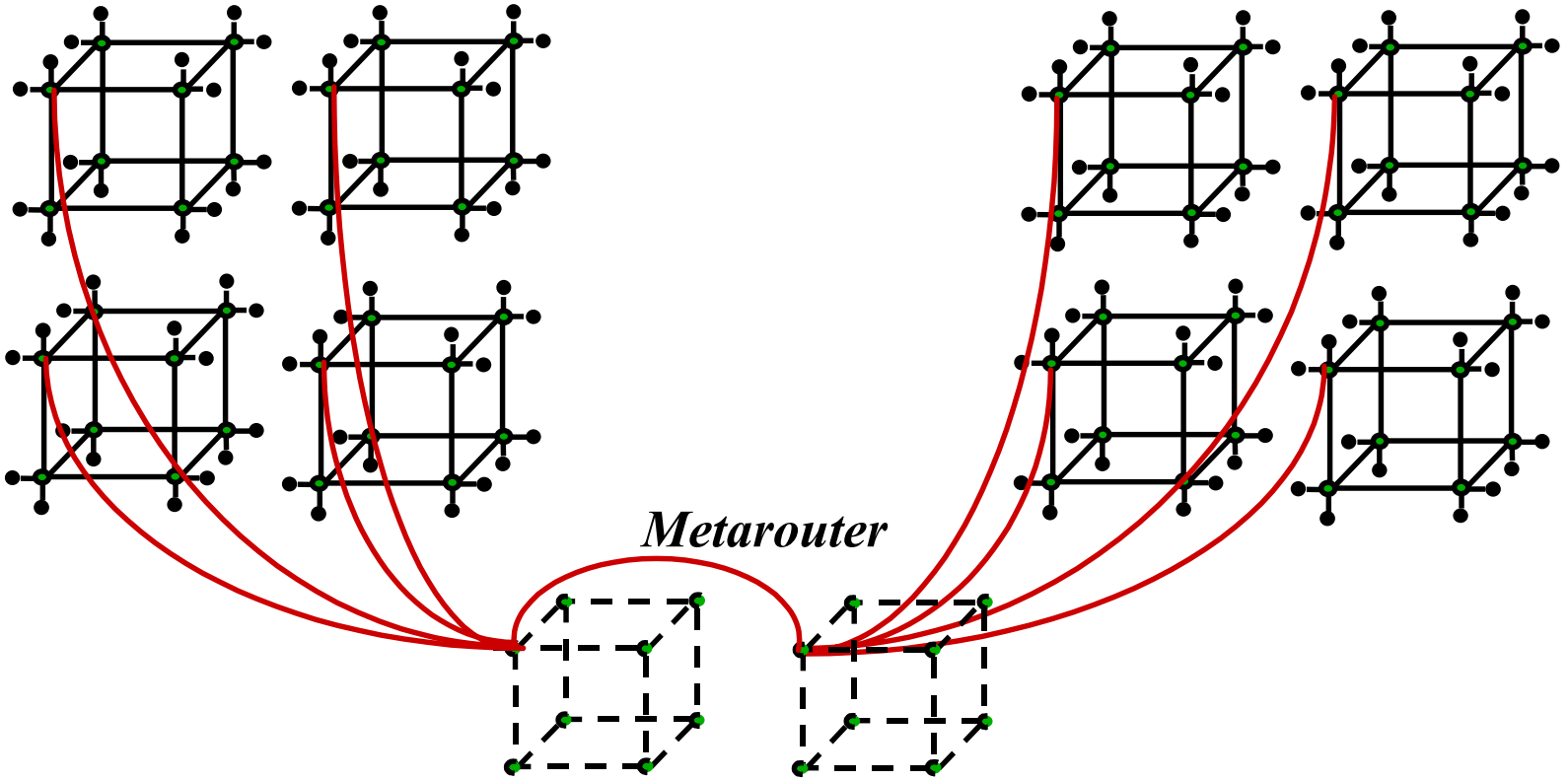


256p

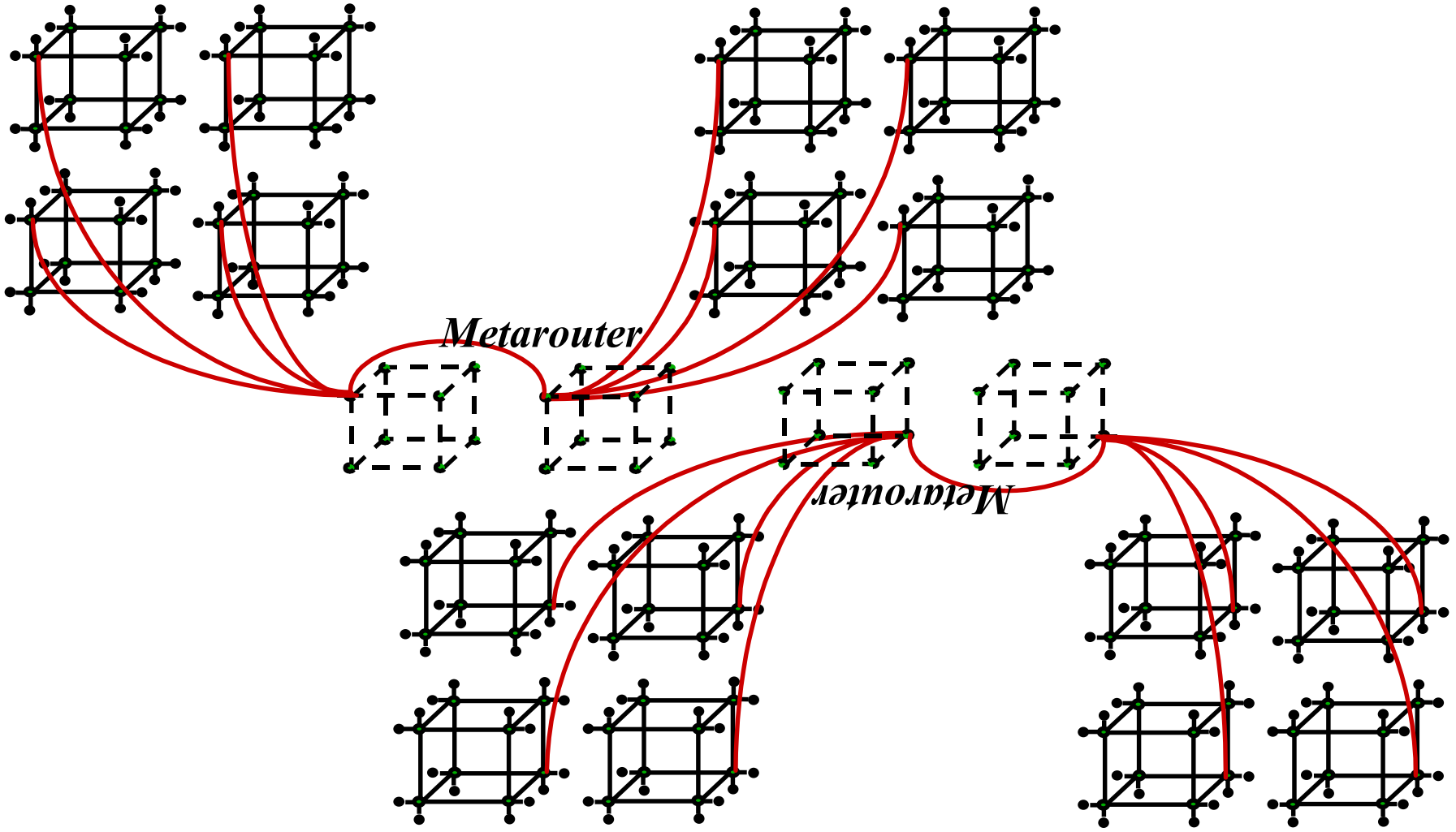


512p

256p Single System Image



512p Single System Image (NUMA Effects Worsen….)

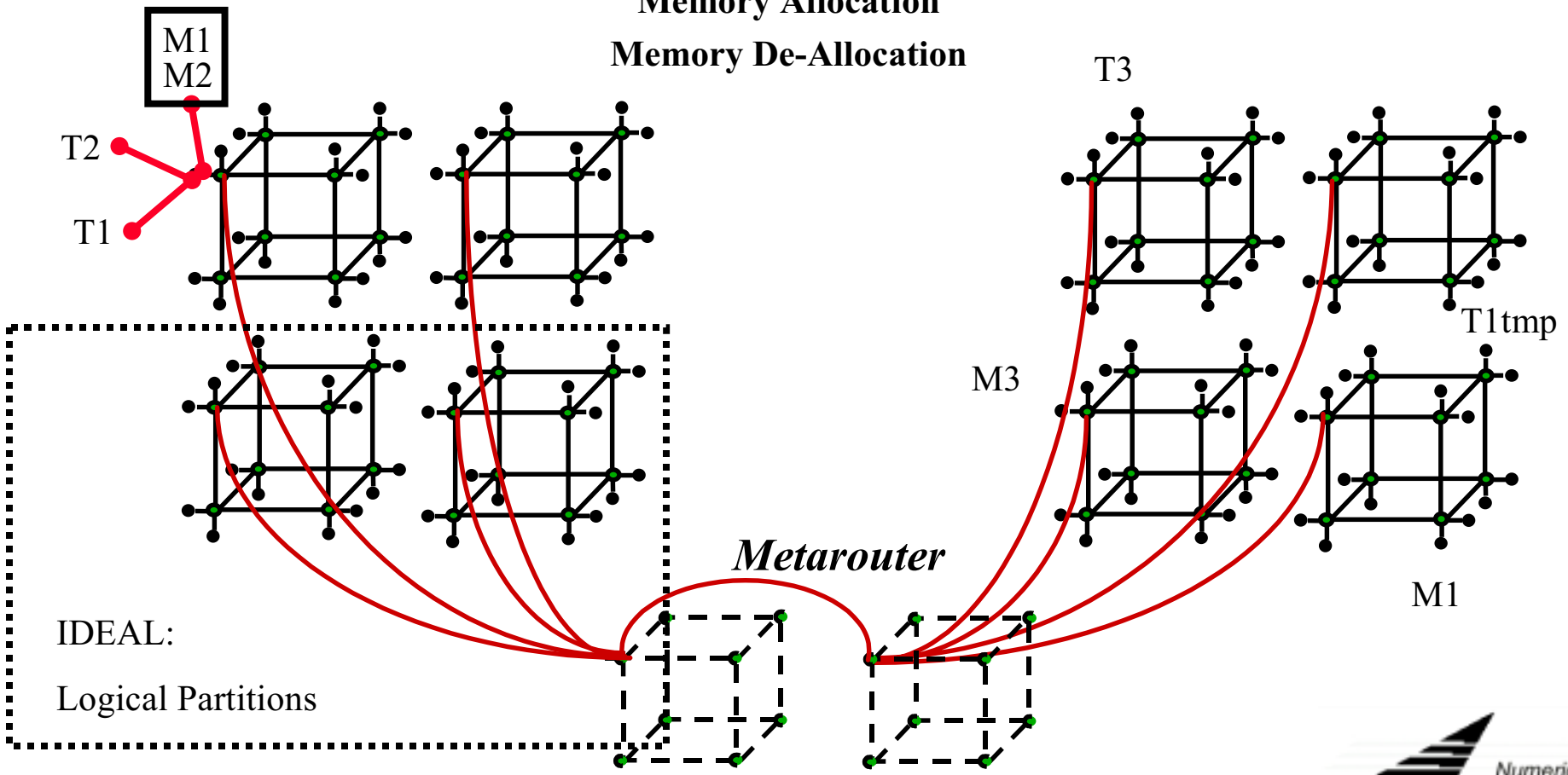


NUMA Aware Operating System

IRIX lacked sufficient control over

Thread Affinity
Thread Location
Memory Allocation
Memory De-Allocation

NUMA effects require that these be dealt with

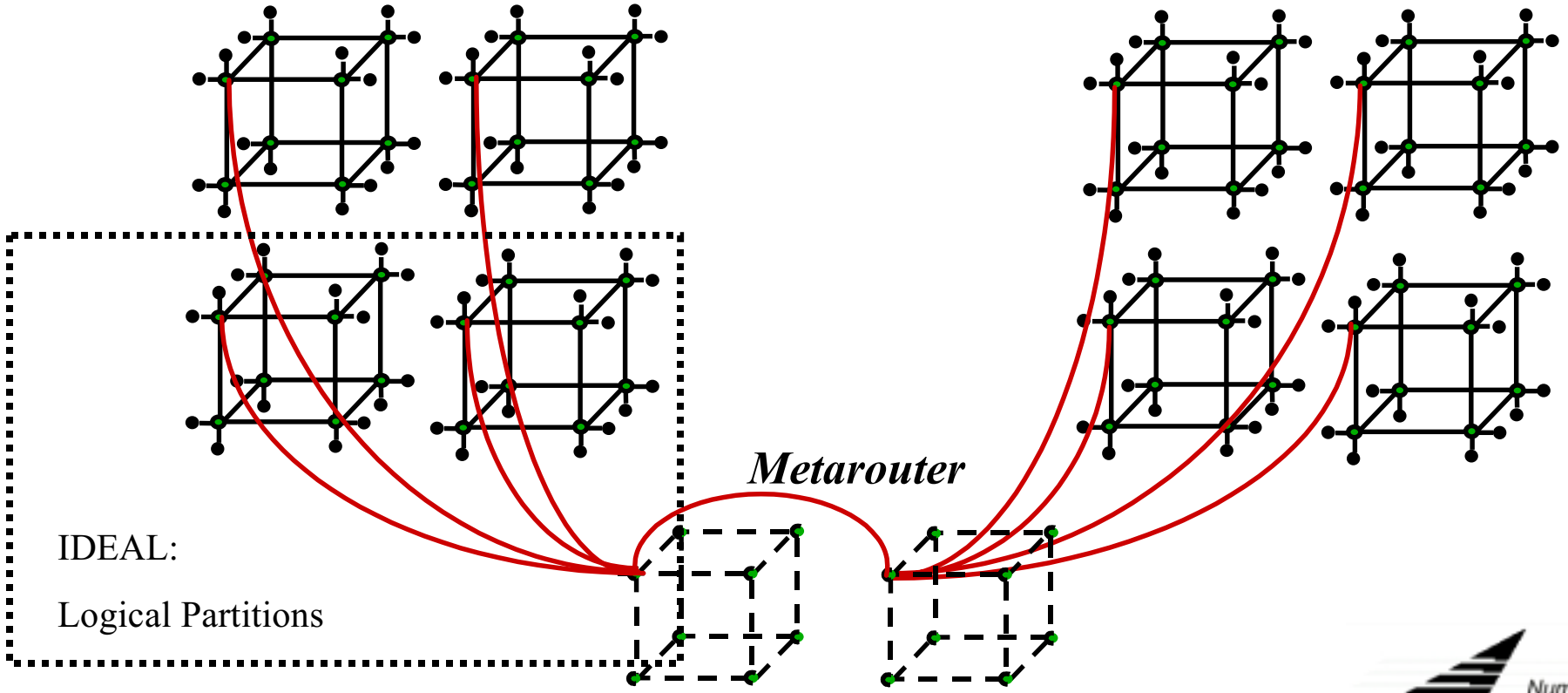


NUMA Aware Operating System

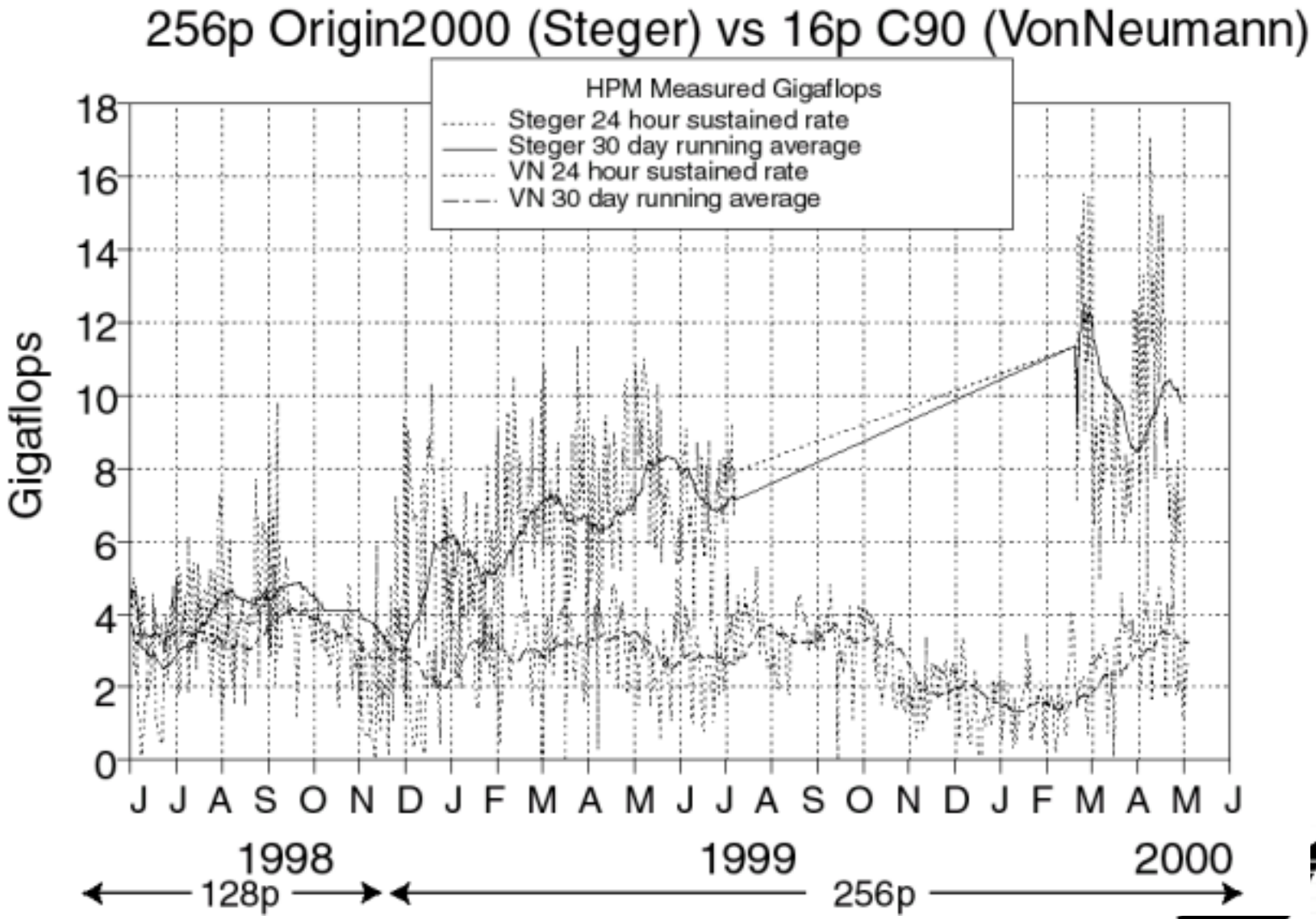
CPU Sets

Memory Limited CPU Sets

Thread Affinity (MLD Sets work)



HPM - C90/16 vs. O2K/256p



Focus on Parallelism

Parallelism is the key to performance on any system manufactured today. If you don't scale to hundreds of CPUs, you won't get to the 50-100 GFLOPS you need today to stay competitive in high end computing. Parallelism is being aggressively pursued on two fronts.

- **Message Passing Interface (MPI)**
  - Many identical processes are spawned (separate a.out's)
  - All communication is performed via explicit "messages"
  - User provides all parallel decomposition/code modification
  - Typically fine-grain
  - Can use compiler parallel multi-cpu nodes but rare

- **Shared Memory Parallelism (OpenMP)**
  - Many lightweight threads are spawned (one a.out)
  - Threads communicate information via memory references
  - Compiler generates parallel code with some user directives
  - Compiler parallelism at loop and routine levels

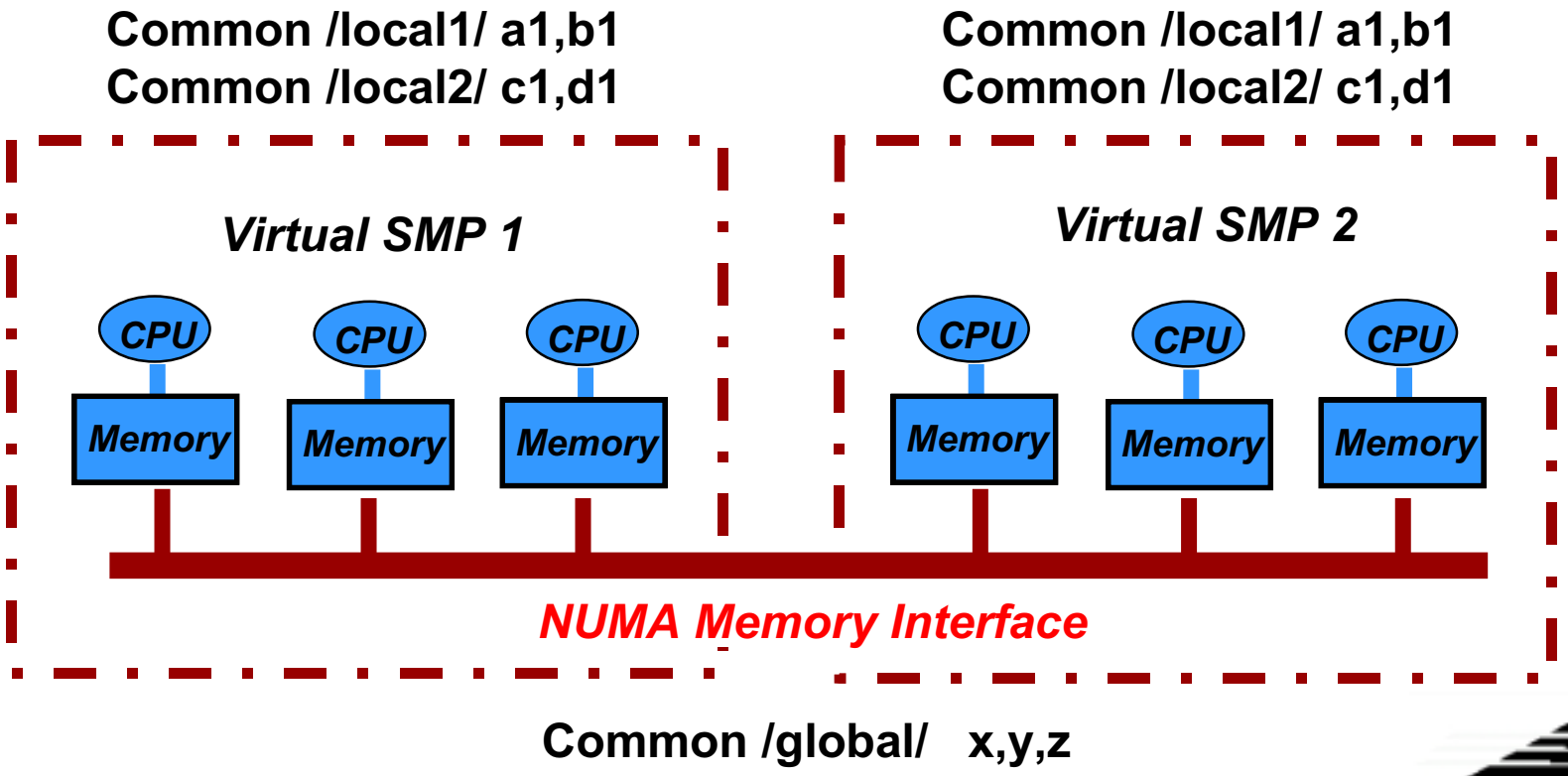
What is Shared Memory MLP?

Shared Memory Multi-level Parallelism (MLP) is the utilization of multiple levels of parallelism within an application executing on a NUMA based system architecture in order to increase its parallel efficiency during execution. It is an open system design and has the following attributes:

- Usually two levels of parallelism
- Coarse grained parallelism provided by Unix forked processes
- Fine grained parallelism provided by the compiler at loop level
- No messaging - communication by Unix shared memory arenas
- Targeted for the new large CPU count NUMA SMP systems
- Method can also execute across clusters

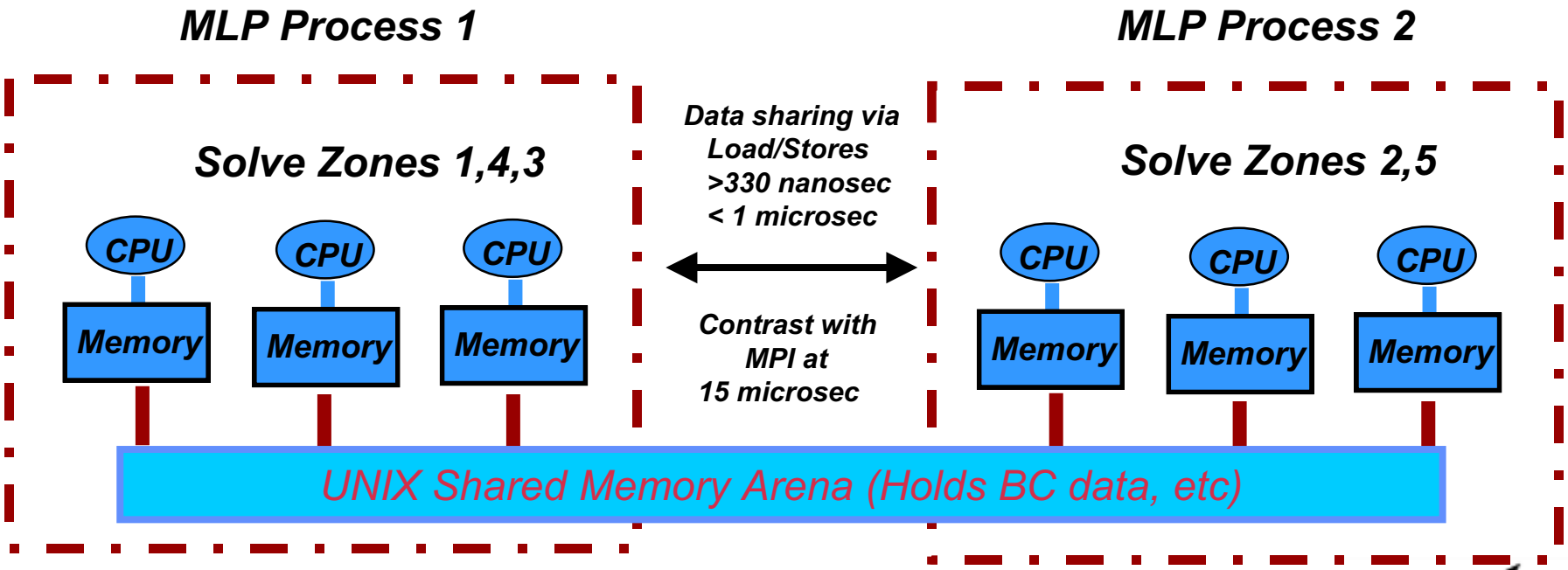
Mapping MLP onto the Origin 2000

The diagram below is a graphical description of how user common blocks are mapped onto the system. In essence each virtual SMP executes a single a.out. Any data the user wants to share across a.out's is declared in the shared memory arena, and essentially appears as a common variable seen by all.



Shared Memory MLP Organization for Origin 2000

This diagram provides a graphical depiction of the application of the MLP processes onto the Origin 2000 hardware. Each process is assigned a small number of CPUs that will be used to perform fine grained loop level parallelism. The CPU count is variable depending on the load in each MLP process. It is also dynamically changeable during a run to allow dynamic load balancing to increase system efficiency. The shared memory arena is simply a UNIX construct that allows each process to directly address the memory of the other MLP processes.



Overflow w/Test Case

This case is a simulation of a full aircraft configured for landing. It is a 35 million point problem distributed across 160 grids of widely varying size. The problem is interesting for several reasons. They are:

- It is one of the largest problems ever solved at NAS
- It requires hundreds of dedicated C90 CPU hours per run
- It fully stress tests the MLP code for correctness and performance
    - Load balance is a major issue with this problem
    - Cache issues are important (some grids break cache)
- It fully tests the Origin system hardware and software components
- The Overflow code is widely used within and outside of NASA

Typical Timing Profile for OVERFLOW

```
-------------------------------------------------------------------------
Profile listing generated June, 1999
-------------------------------------------------------------------------
Sorted in descending order by the number of samples in each procedure.
-------------------------------------------------------------------------

samples   time (%)     cumulative time(%)  subroutine

   3446    34s(  5.3)        34s  (  5.3)   SPE2JK  (BM61.x:SPE2JK.f)
   3433    34s(  5.3)        69s  ( 10.7)   G2SL    (BM61.x:G2SL.f)
   3408    34s(  5.3)   1.0e+02s  ( 16.0)   S2GL    (BM61.x:S2GL.f)
   2545    25s(  3.9)   1.3e+02s  ( 19.9)   VFLL    (BM61.x:VFLL.f)
   2393    24s(  3.7)   1.5e+02s  ( 23.6)   SCALE   (BM61.x:TSCALE.f)
   2184    22s(  3.4)   1.7e+02s  ( 27.0)   TKL     (BM61.x:TKL.f)
   2167    22s(  3.4)   2.0e+02s  ( 30.4)   SPE2JL  (BM61.x:SPE2JL.f)
   2047    20s(  3.2)   2.2e+02s  ( 33.5)   FLCJ    (BM61.x:FLCJ.f)
   1855    19s(  2.9)   2.3e+02s  ( 36.4)   v1pen   (BM61.x:v1pen.f)
   1747    17s(  2.7)   2.5e+02s  ( 39.1)   TKINV   (BM61.x:TKINV.f)
   1639    16s(  2.5)   2.7e+02s  ( 41.7)   NPINV   (BM61.x:NPINV.f)
   1587    16s(  2.5)   2.8e+02s  ( 44.1)   EIGCJ   (BM61.x:EIGCJ.f)
   1531    15s(  2.4)   3.0e+02s  ( 46.5)   COPY    (BM61.x:COPY.f)
   1521    15s(  2.4)   3.2e+02s  ( 48.9)   v1pen3  (BM61.x:v1pen3.f)
   1228    12s(  1.9)   3.3e+02s  ( 50.8)   BCCHAR  (BM61.x:BCCHAR.f)
   1183    12s(  1.8)   3.4e+02s  ( 52.6)   QADDS   (BM61.x:QADDS.f)
   1078    11s(  1.7)   3.5e+02s  ( 54.3)   CARK    (BM61.x:CARK.f)
   1076    11s(  1.7)   3.6e+02s  ( 55.9)   STFLL   (BM61.x:STFLL.f)
   1014    10s(  1.6)   3.7e+02s  ( 57.5)   FLCL    (BM61.x:FLCL.f)
```

This table is a small segment from the total profile showing the time spent in the major routines of OVERFLOW. It shows that the code has no major hot spots, with the work evenly distributed across many routines.

This flat profile makes it very difficult to optimize the code in the traditional sense as a large number of routines must be modified before any significant gains in run time can be had.

The best hope for codes of this type is to add modifications that allow efficient parallel scaling at the coarse level and execute on large numbers of CPUs.

35M Point Airplane Problem: BALANCE Routine Output

```
Problem Name    :35M Airplane
--------------------------------

Number of Zones :          160
Largest Zone    :      1414140
Smallest Zone   :        11475
```

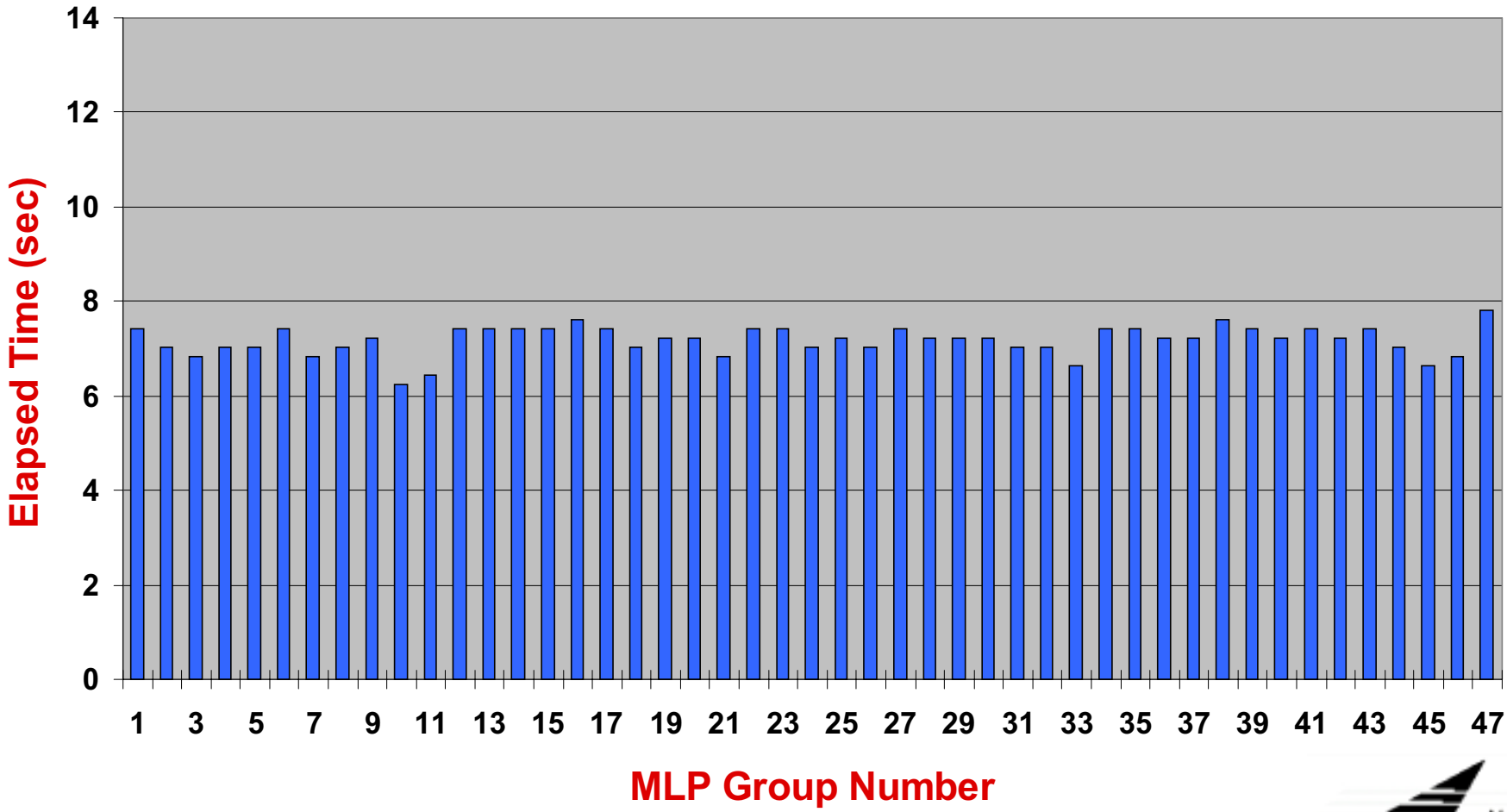
Load Balancing is automatic and dynamic

```
Summary of Load Balance Analysis:                          Group Number:  1
-------------------------------------------------------------------------
Grid     JD    KD    LD   Iter    Points  Weighted   %Volume     CPUs
-------------------------------------------------------------------------
   6    111   140    91      1   1414140   1414140      3.99        6
  89     65    35    51      1    116025    116025      0.33        6
 130     81    19    45      1     69255     69255      0.20        6
  31     36    31    41      1     45756     45756      0.13        6
  65     15    41    51      1     31365     31365      0.09        6
  25     13    23    45      1     13455     13455      0.04        6
-------------------------------------------------------------------------
Totals:                        1689996   1689996      4.76        6

Summary of Load Balance Analysis:                          Group Number: 21
-------------------------------------------------------------------------
Grid     JD    KD    LD   Iter    Points  Weighted   %Volume     CPUs
-------------------------------------------------------------------------
 116     65    81    85      1    447525    447525      1.26        6
 158    189    41    49      1    379701    379701      1.07        6
 157     37   161    45      1    268065    268065      0.76        6
  27    303    16    45      1    218160    218160      0.61        6
  44     81    33    51      1    136323    136323      0.38        6
 118     27    61    51      1     83997     83997      0.24        6
  32     48    31    41      1     61008     61008      0.17        6
 110     53    31    31      1     50933     50933      0.14        6
  54     14    41    51      1     29274     29274      0.08        6
  72     21    25    45      1     23625     23625      0.07        6
-------------------------------------------------------------------------
Totals:                        1698611   1698611      4.79        6

BALANCE: Total point count :   35483075
BALANCE: Total CPU   count :        126
```

OVERFLOW: 35M Point Airplane Problem on 256 CPUs (steger)
Elapsed Time by MLP Group For One Time Step
(91% Aggregate CPU Efficiency)



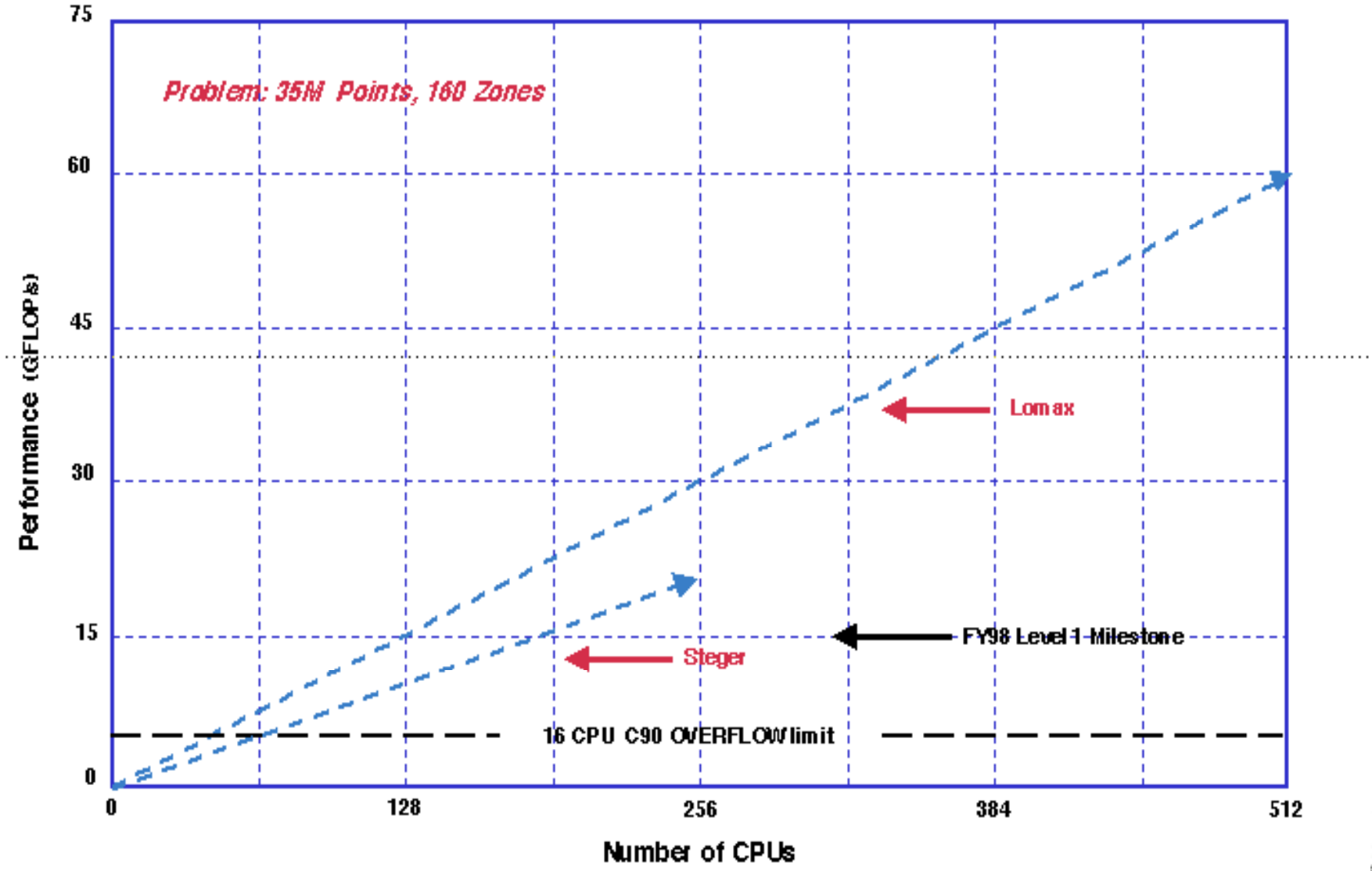
Dynamic Load Balancing; LEVLER Routine

Many CFD codes may incur load imbalance as the result of new and different work occurring as the simulation progresses (ie clouds forming in weather models, etc). The MLP technique offers a simple way of dynamically balancing the load as a function of time. It is:

- **Take a time step**
- **Find the MLP group that runs the longest**
- **Get it to run faster by:**
  - **Moving a zone to a faster group**
  - **Moving a CPU in from a faster group**
- **This switch has virtually no overhead as data is shared**

This ability to quickly (in microseconds) dynamically load balance is vital for many NASA production codes. MLP makes this process trivial, a major advantage over the MPI approach.

OVERFLOW-MLP Performance vs CPU Count
Steger (250MHz O2K/256p) Lomax (300MHz O2K/512p)

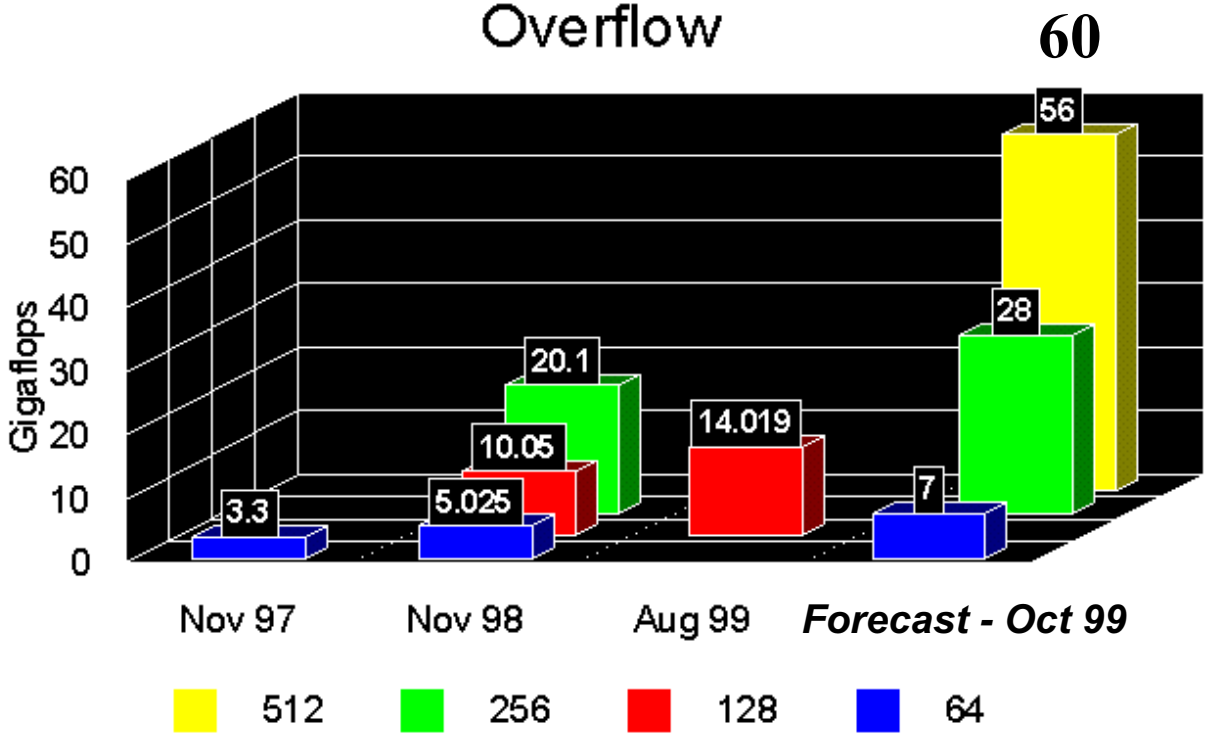


MLPlib

The MLPlib routines for scalable parallel execution support are:

- *Subroutine INITMEM(numbyt)*
  - *The INITMEM routine sets up a UNIX shared memory arena consisting*
  - *of numbyt bytes to be used by all subsequently spawned processes.*
- *Subroutine GETMEM(xarray,xpoint,numxbyt)*
  - *The GETMEM routine allocates numxbyt bytes to the xarray variable*
  - *resident in the shared memory arena. The xarray data will be visible to all*
  - *MLP processes using the shared memory arena.*
- *Subroutine FORKIT(numpro,nowpro)*
  - *The FORKIT routine spawns a total of numpro processes for the job*
  - *including the current one in the total. Nowpro is returned, and is the*
  - *current process id (0-n).*
- *Subroutine BARRIER(numpro)*
  - *The BARRIER routine waits until numpro processes have hit the*
  - *barrier, then all drop through.*

Extension of OpenMP - MLPlib



MLP Observations

- Ease of Use
- Simplifies validation
- Higher degree of parallelism
- Able to extract performance
- Results often agree with C90 versions (same algorithm/source)
- Much easier to migrate legacy codes
- Viable alternative to MPI

Bottom Line - MLP

Big win in Performance and Simplicity

MTTI - C90/16

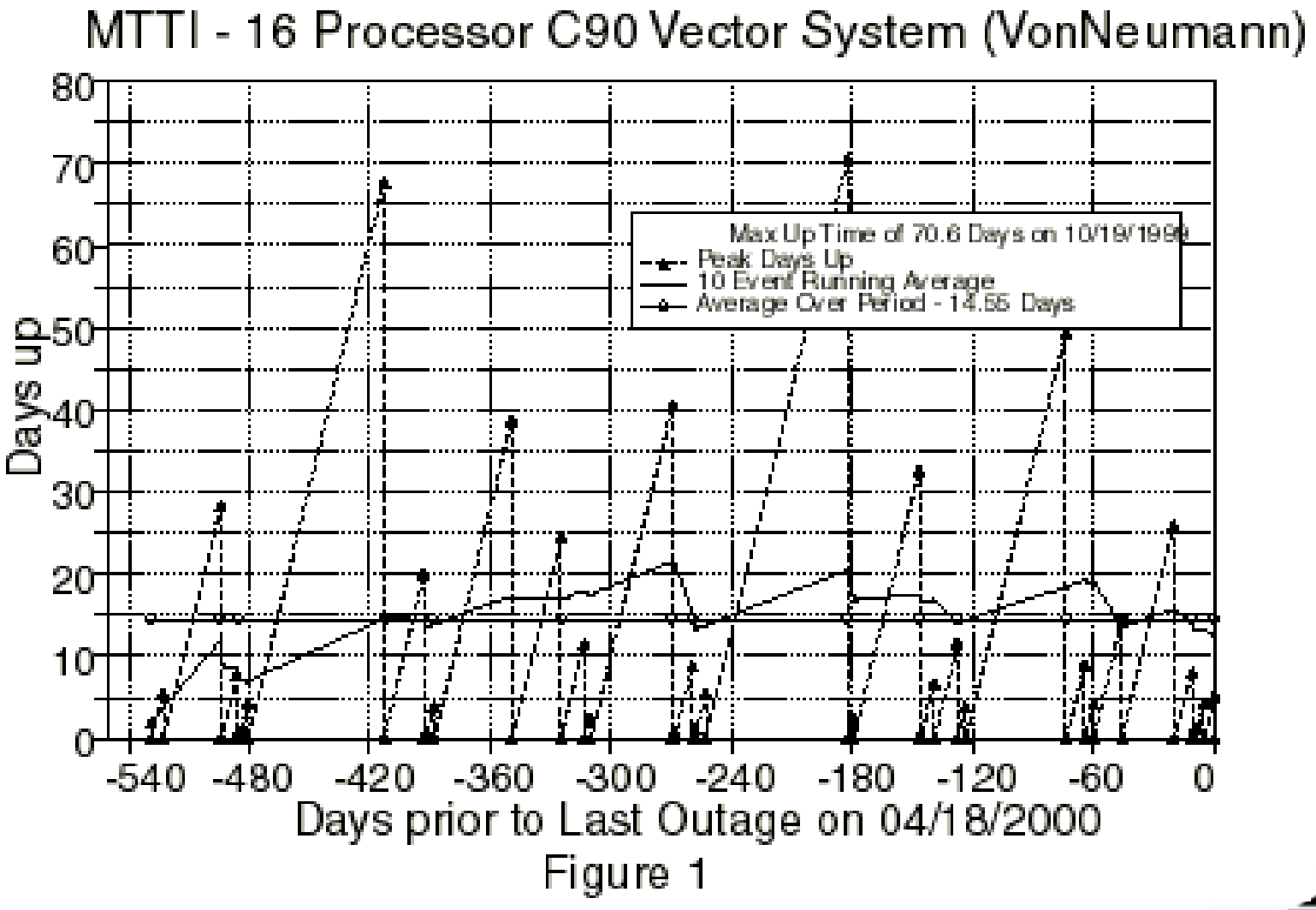


Figure 1

MTTI - O2K/256p

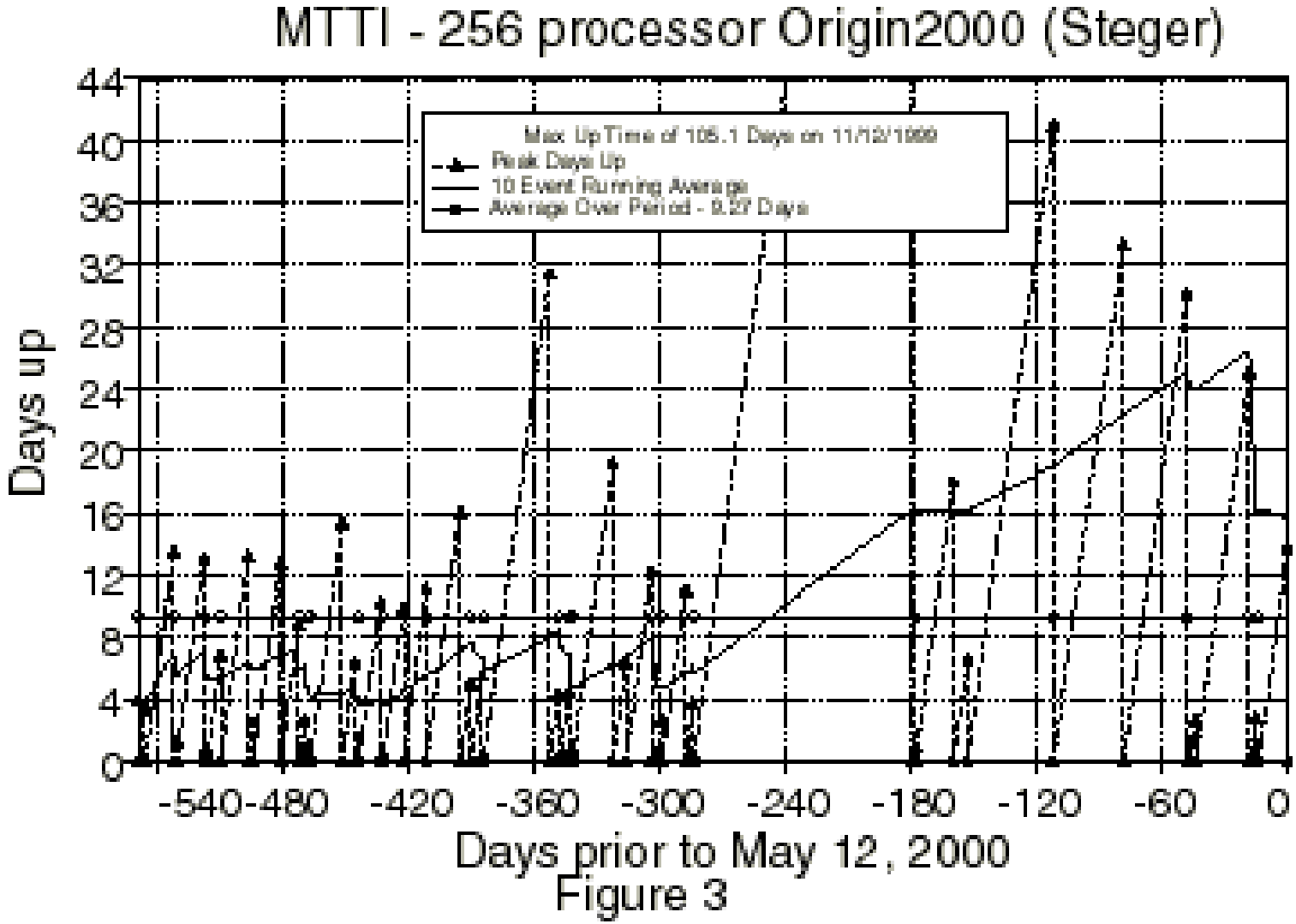


Figure 3

MTTI - O2K/512p

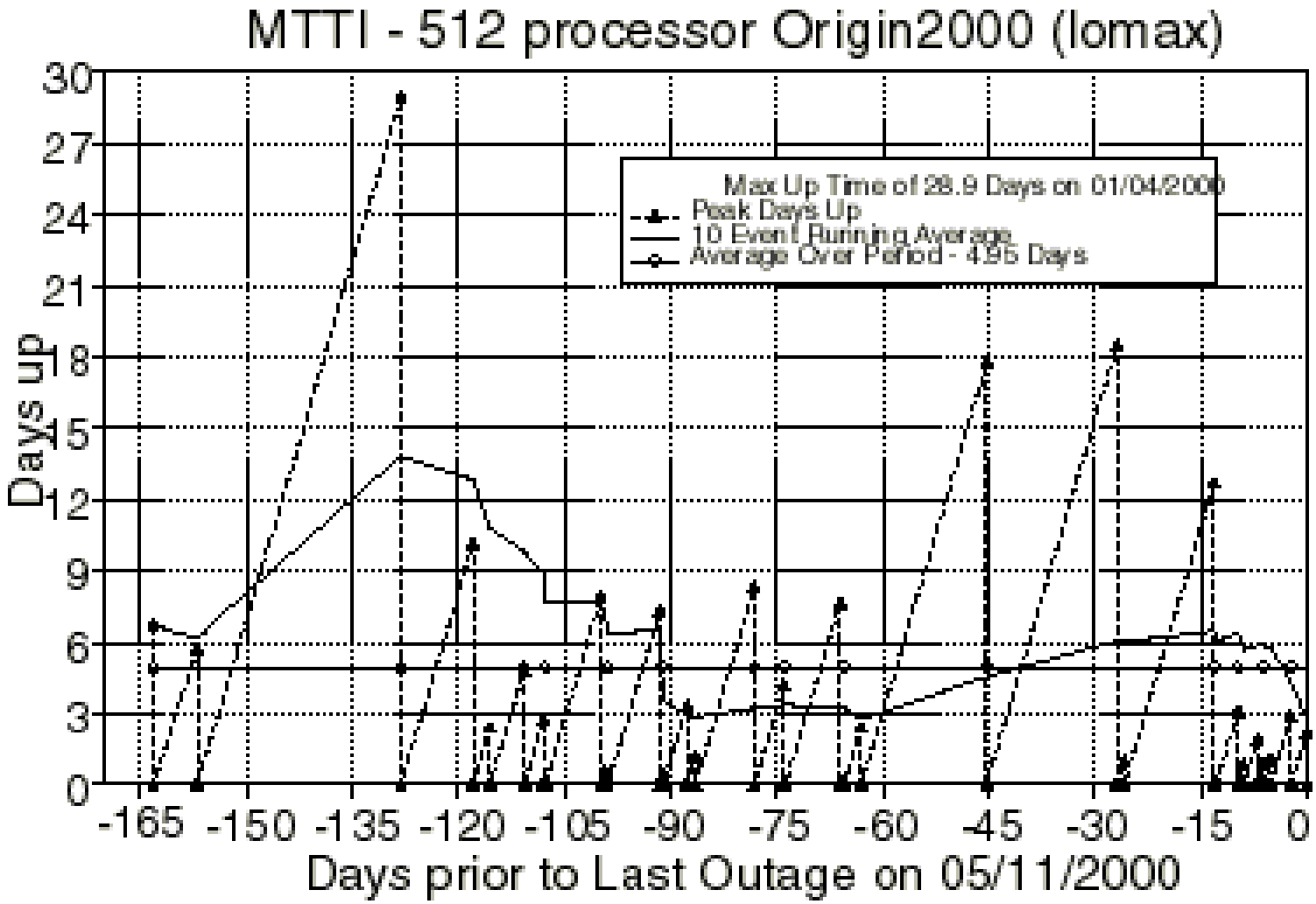


Figure 4

MTTI - C90/16 vs. O2K/256 vs O2K/512

	16p C90 Vector (vn)				256p Origin2000 (steger)				512p Origin200 (lomax)			
	Total #Days	. %Time	Evnt #	MTTI Days	Total #Days	. %Time	Evnt #	MTTI Days	Total #Days	. %Time	Evnt #	MTTI Days
Software Outage	1.08	0.4%	6	47.4	0.46	0.2%	5	56.6	1.28	0.8%	29	5.9
Hardware Outage	3.04	1.1%	13	21.9	2.10	0.7%	8	35.4	0.38	0.2%	6	28.3
Other Caused Outage	-	0.0%	10	28.5	0.07	0.0%	1	282.9	-	0.0%	1	170.0
Tape Caused Outage	0.35	0.1%	7	40.6	-	0.0%	-	-	-	0.0%	-	-
NFS Caused Outage	-	0.0%	-	-	-	0.0%	-	-	0.03	0.0%	1	170.0
Network Caused Outage	0.06	0.0%	2	142.3	-	0.0%	1	282.9	-	0.0%	-	-
Unsched Facility Shutdown	2.22	0.8%	4	71.1	0.91	0.3%	4	70.7	-	0.0%	-	-
Sched Dedicated Time	2.06	0.7%	14	20.3	1.39	0.5%	17	16.6	4.24	2.5%	34	5.0
Sched Preventative Maint	1.85	0.7%	15	19.0	-	0.0%	1	282.9	0.10	0.1%	1	170.0
Sched Facility Shutdown	4.46	1.6%	4	71.1	3.97	1.4%	3	94.3	3.64	2.1%	2	85.0
System up with problem	18.75	6.6%	604	0.5	0.57	0.2%	24	11.8	0.53	0.3%	15	11.3
System up without problem	250.67	88.1%	.	-	273.38	96.7%	.	-	159.79	94.0%	.	-
Total of All Outages	284.54	.	75	3.8	282.85	.	40	7.1	169.99	.	74	2.3
Total All Unsched Outages	284.54	.	42	6.8	282.85	.	19	14.9	169.99	.	37	4.6
Total of HW/SW Outages	284.54	.	19	**15.0**	282.85	.	13	**21.8**	169.99	.	35	**4.9**

Accomplishment
Demonstrated Scaling on Real Application

Year	Machine	#CPUs	Cost (Million$)	Optimized Code Performance (gigaflops)	Price Performance (K$/Gigaflop)	Performance to C90/16	Price Perf to C90/16
1994	C90	16	$ 47.70	4.6	$ 10,370	1.0 X	1.0 X
1996	SX4	32	$ 17.90	22.0	$ 814	4.8 X	12.7 X
1998	O2K	256	$ 9.70	20.1	$ 483	**4.4 X**	**21.5 X**
1999	O2K	512	$ 16.50	60.0	$ 275	**13.0 X**	**37.7 X**

*NOTE: 1996 SX4 performance numbers are from 32 simultaneous ARC3D runs while C90 and O2K numbers are from a single 30M point Overflow run in 1999.

Lomax: 512 Processor Origin2000

CPUs

- 512 MIPS R12000
  - 300 MHz CPUs
- 600 MFLOP/s per CPU Peak
- 307 GFLOPS total
  - (19 x C90) Peak
- 8 MByte cache per CPU (4 GByte total)

Memory

- 192 GB main memory
  - (24 x C90)
- Hypercube interconnect (9 hops)

Disk

- 2 TB FC Raid disk sub-system

System Software

- Develop OS true single system image
- Single XFS File System
- Compiler parallel loops 512 CPUs wide

Xtreme ccNUMA

Cost/performance OVERFLOW-MLP

- ***Cost/perf O2K/C90 = 37.7x***
- ***Cost/perf O2K512 to O2K256 = 1.6x***
- ***Performance over 256p = 3x***
- ***Performance over C90/16 = 13x***

Impact of Large Scale Capability

- Laura - Langley Aero-thermal Upwind Relaxation Algorithm
  - CFD
  - Chemistry - includes catalytic properties of wing material
- X37 - Drone Prototype
  - Designed to be dropped out of Shuttle for re-entry
  - Cannot be physically modeled (e.g. wind tunnels are out)
  - Run on Dedicated 256p and 512p systems
- Capability allowed NASA to discover design flaw
  (thermal overload >5000^{O} On wing leading edge - catastrophic failure condition)
- What was months is now days
- Availability of 256p/512p in Fall/Winter 1999 made this possible - Time Matters