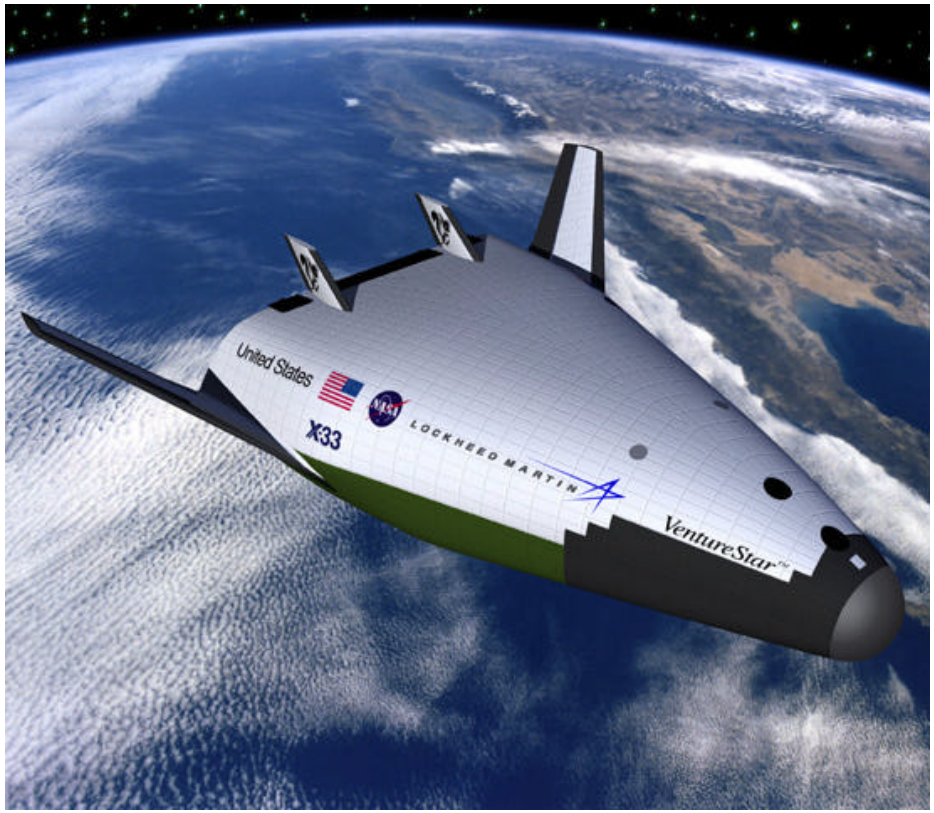
Information Sciences & Technology
Ames Research Center



Tera Scale Systems and Applications

Bob Ciotti
Tera-Scale Application Group Lead
NASA Advanced Supercomputing Division
NASA Ames Research Center
Moffett Field, CA 94035-1000
Ciotti@nas.nasa.gov

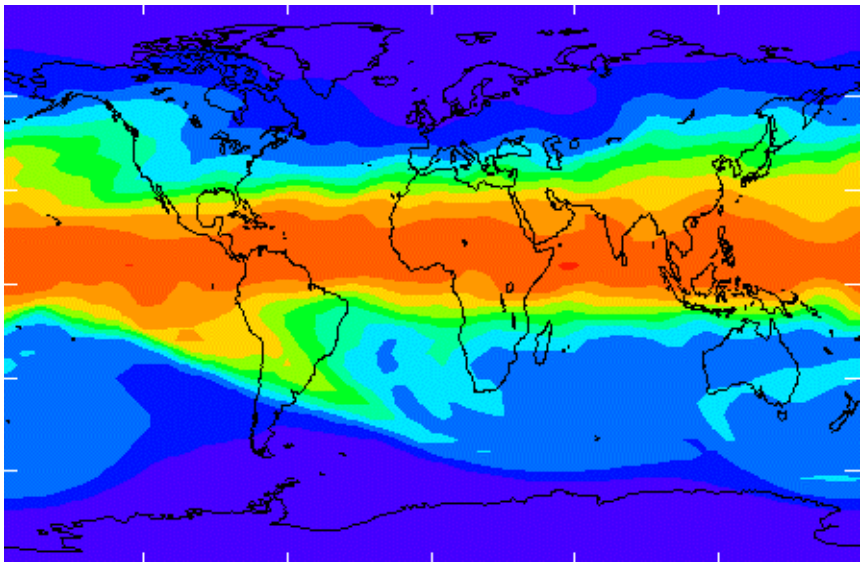
***N**ASA*
***A**dvanced*
***S**upercomputing*

NASA's Computational Challenges

- Aeronautics
  - X-Vehicle Development
  - Shuttle Upgrades
  - Airspace Control and Simulation
- Astrobiology
  - Protein Folding and Beyond
- Earth Science
  - Atmospheric Physics
  - Oceanography
- Nano Technology
  - Chemistry
- Space Science
  - Stellar Dynamics
  - Chemistry/Spectra
  - Instrument Data Analysis and Support
  - Planetary Modeling

NASA Advanced Supercomputing Facility

	Current SGI Inventory - ~ 20 systems			
	#CPUs	System	Name	
	512	O2k	Lomax	Development
	1024	O3k	Chapman	Development
	8	O2k	Evelyn	Development
	4	O2k	Piglet	Development
	1548	**Subtotal**		
	512	O2k	Lomax2	Production
	256	O2k	Steger	Production
	64	O2k	Hopper	Production
	64	O2k	Dixon0	Production
	64	O2k	Jimpf0	Production
	64	O2k	Jimpf1	Production
	64	O2k	Kalney	Production
	64	O2k	Sunrise	Production
	24	O2k	Turing	Production
	8	O2k	Fermi	Production
	1184	**Subtotal**		
	12	O2k	Lou	Storage
	8	O2k	Lou2	Storage
	4	O2k	This	Storage
	4	O2k	That	Storage
	28	**Subtotal**		
	2760	**Origin CPUs**		



Goal: Production Quality Highly Parallel Supercomputer

- What is a Production Supercomputer?
  - Significantly faster that previous generation
  - Less Expensive
  - Productive Environment for the Users (easy to use)
  - As reliable as the “Gold Standard”

- Have to move into the realm of hundreds or thousands of processors
  - Its all about the interconnect

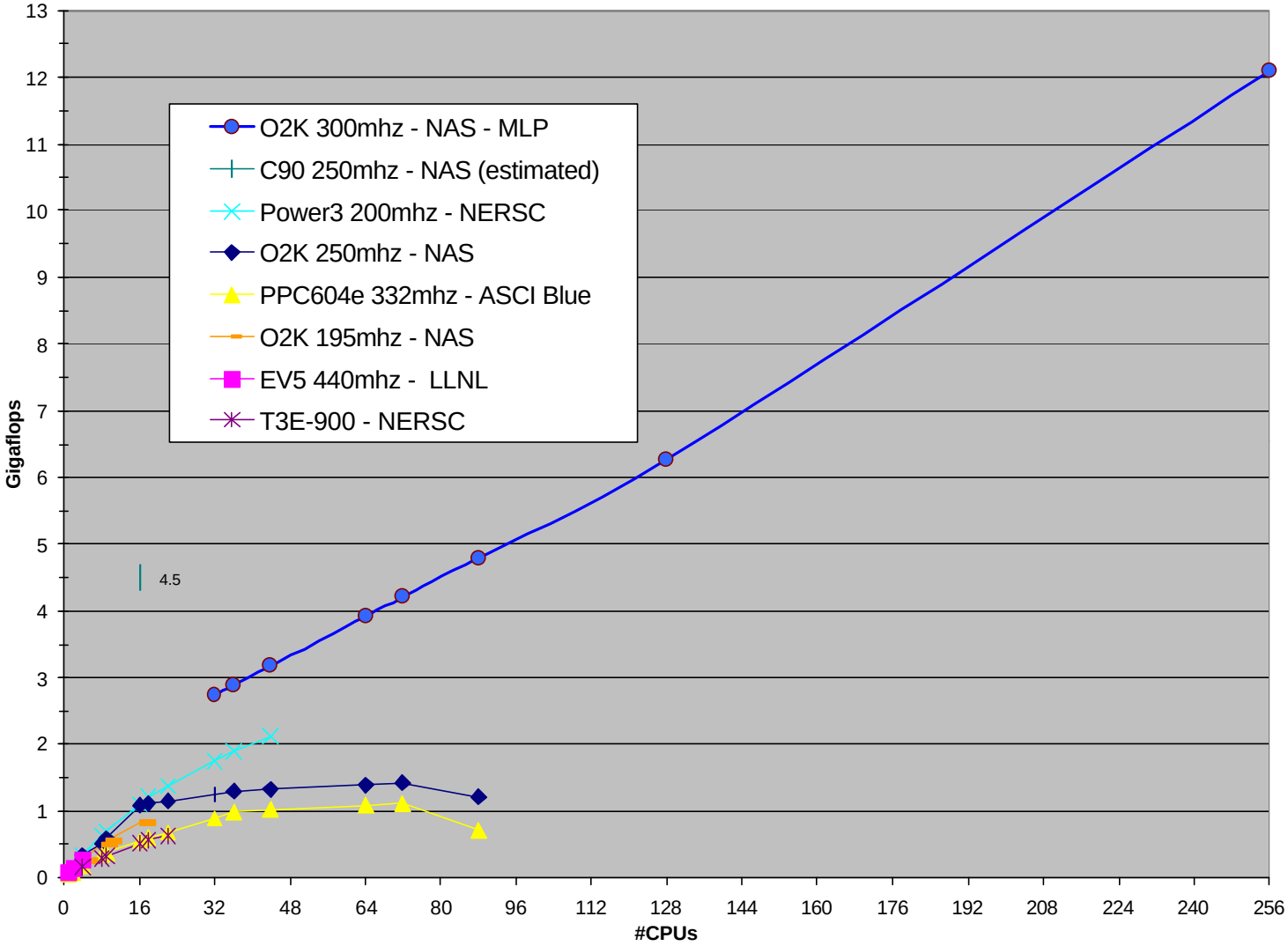
- Many Attempts (these all failed…)
  - Connection Machines - CM2 (performance)
  - Connection Machines - CM5 (performance)
  - Intel - IPSC-860 (performance)
  - Intel – Paragon (performance)
  - IBM - SP2 (performance)
  - SGI - Power Challenge Cluster (reliability)
  - Cray - J90 Cluster (vendor backed out of commitment)

Barriers for Scientists Obtaining a SuperComputing Capability for Their Research

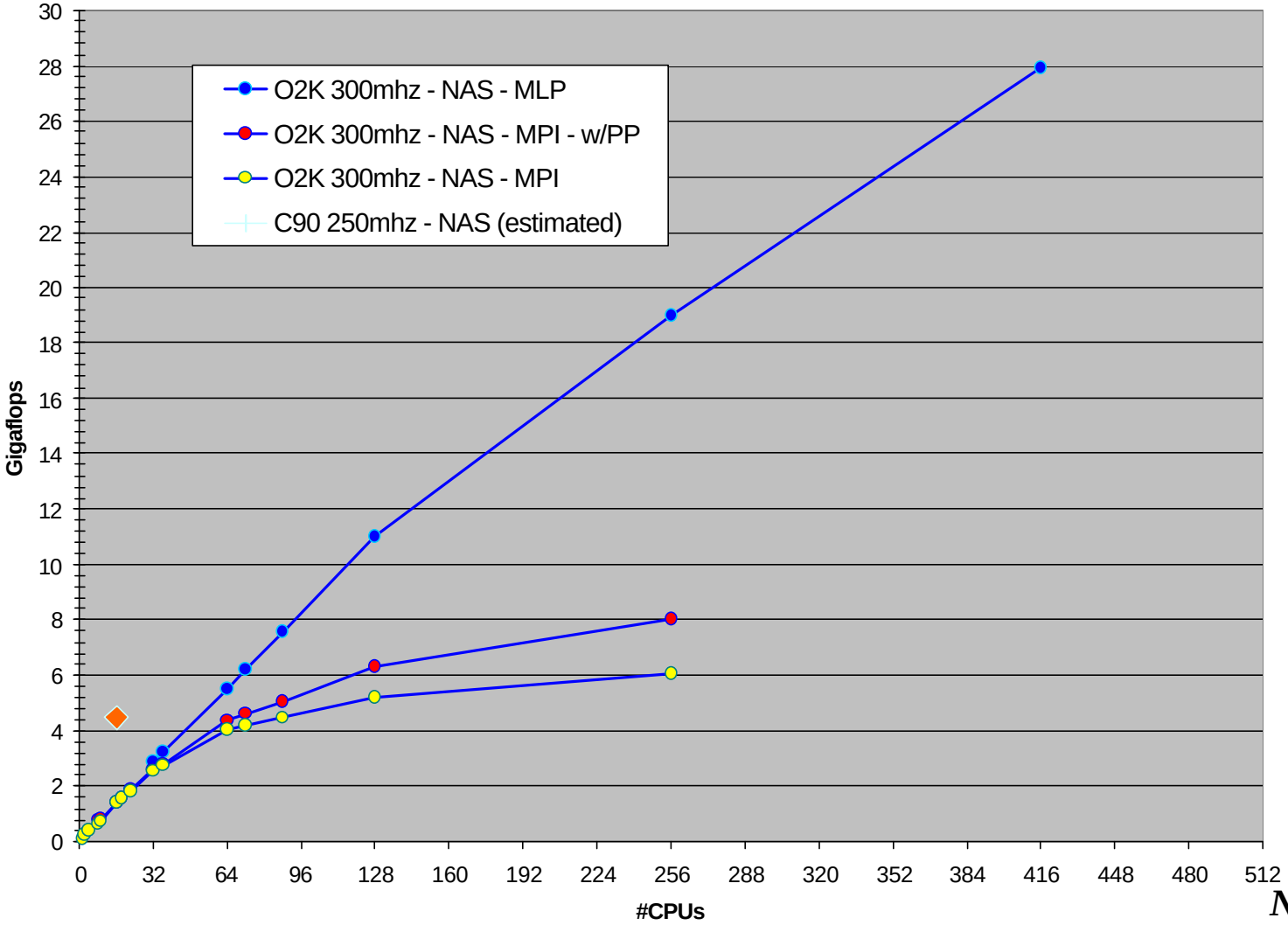
- There are many problems that require supercomputing
- However, its typical that applications may require substantial modifications to achieve even moderate levels of parallelism.
- This can translate into several man years of effort, and when you have done this, you'll likely not run faster that a C90 supercomputer manufactured in 1993.
- Many (most) productive scientists are simply unable to access supercomputing because it either difficult or even not possible to effectively scale their applications.
- For example,
  - Computational Chemistry
  - DAO
  - Molecular Dynamics
  - Space Science

Shared memory can go a long way towards SIMPLIFICATION.

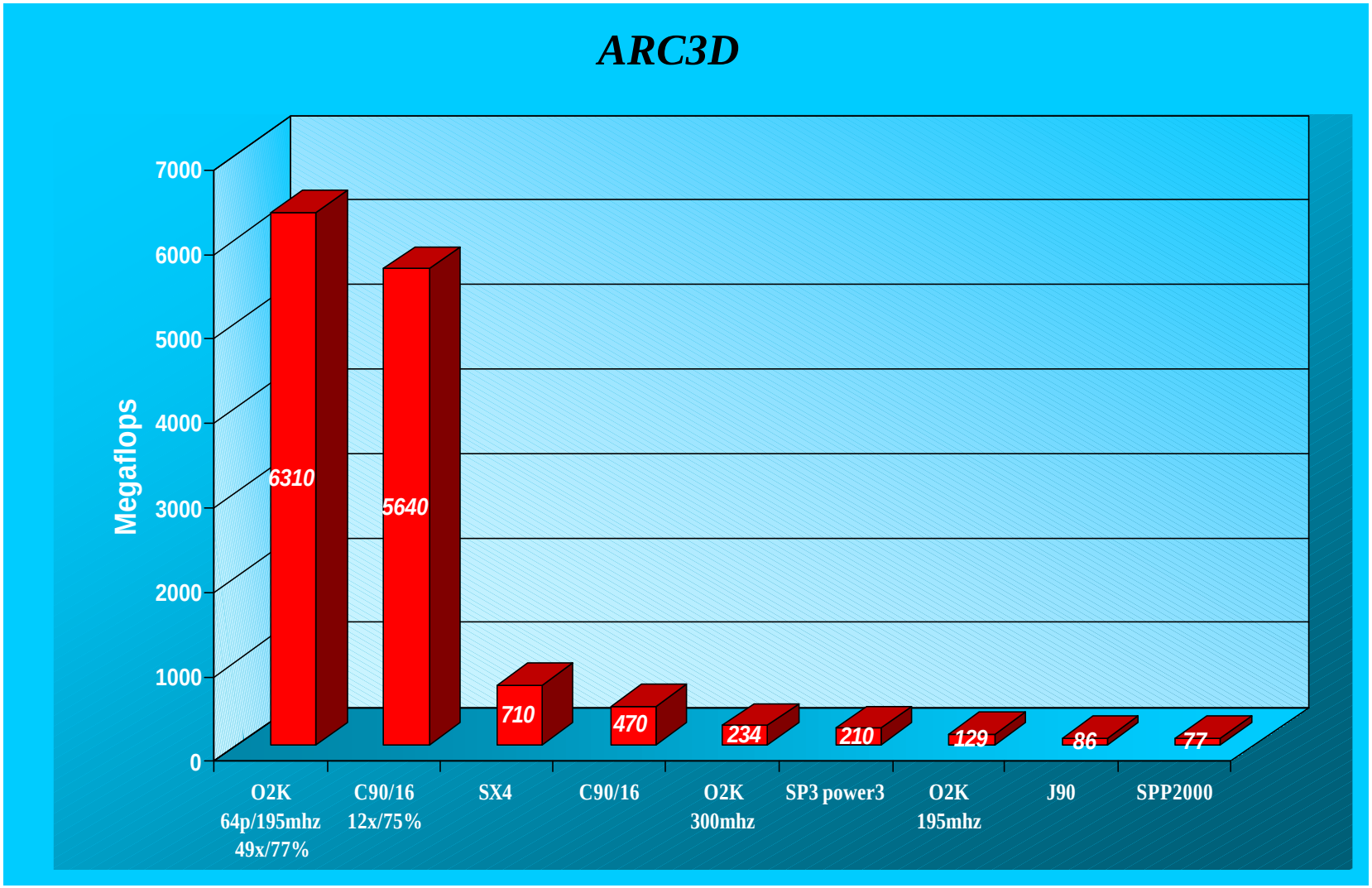
Are we getting sustained performance on parallel systems for our real applications?



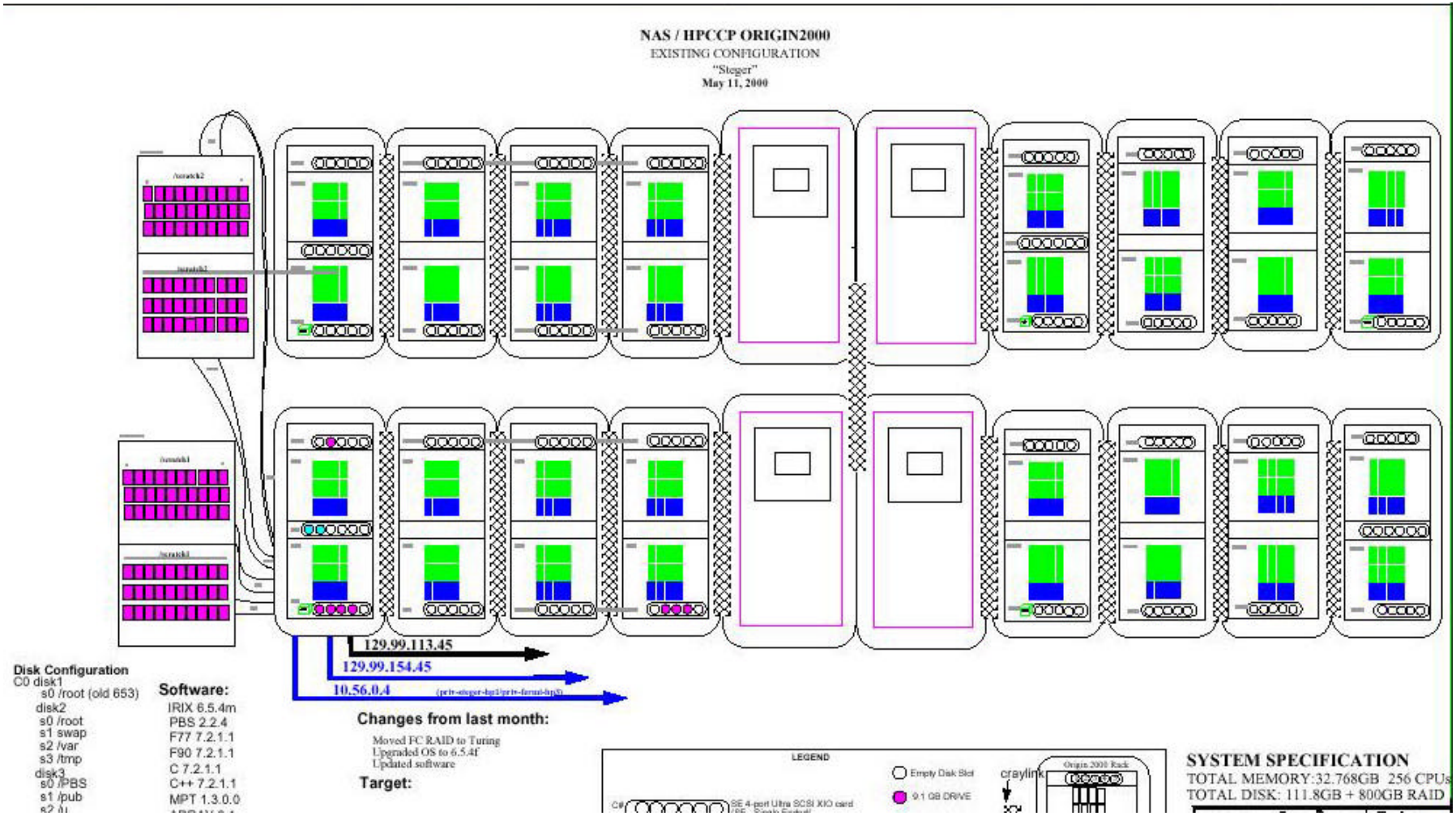
Are we getting sustained performance on parallel systems for our real applications?



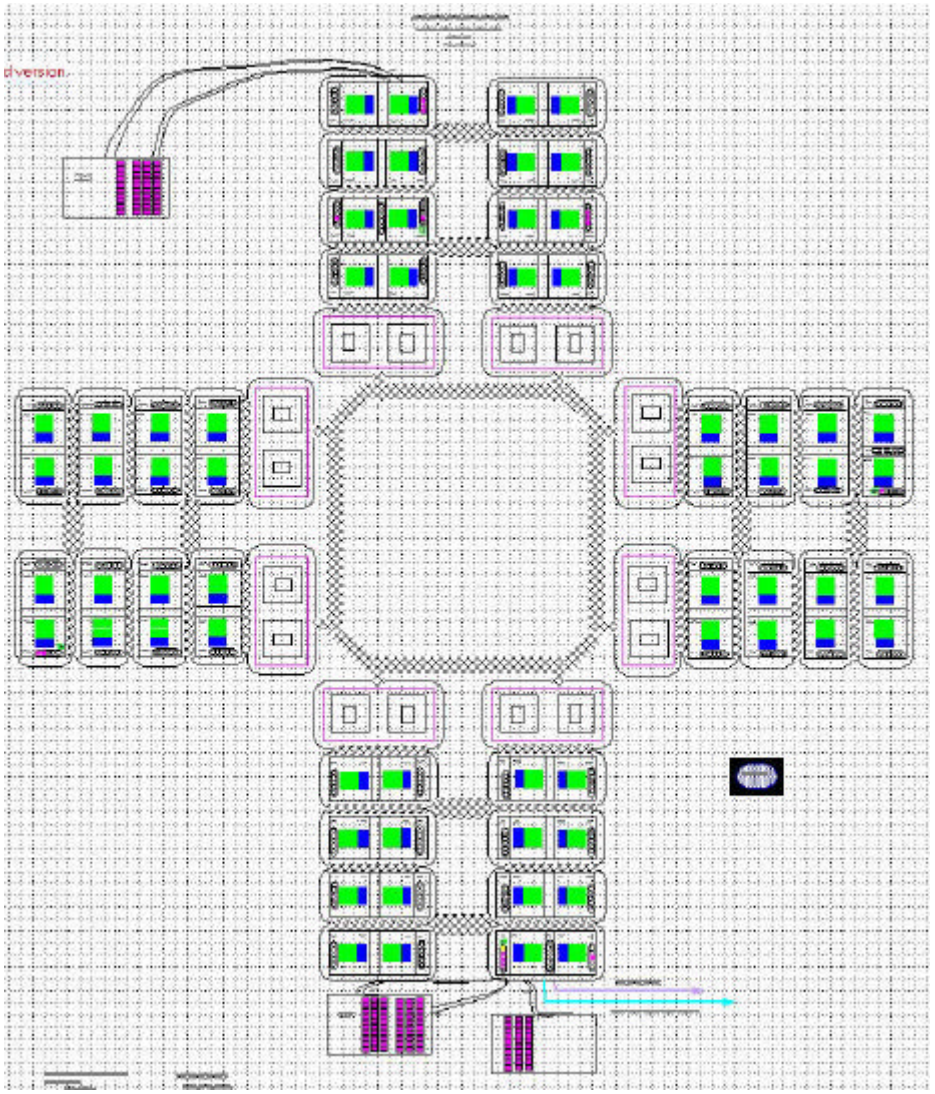
Performance
(mostly from 1996)



256p Origin2000



512p Origin2000



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

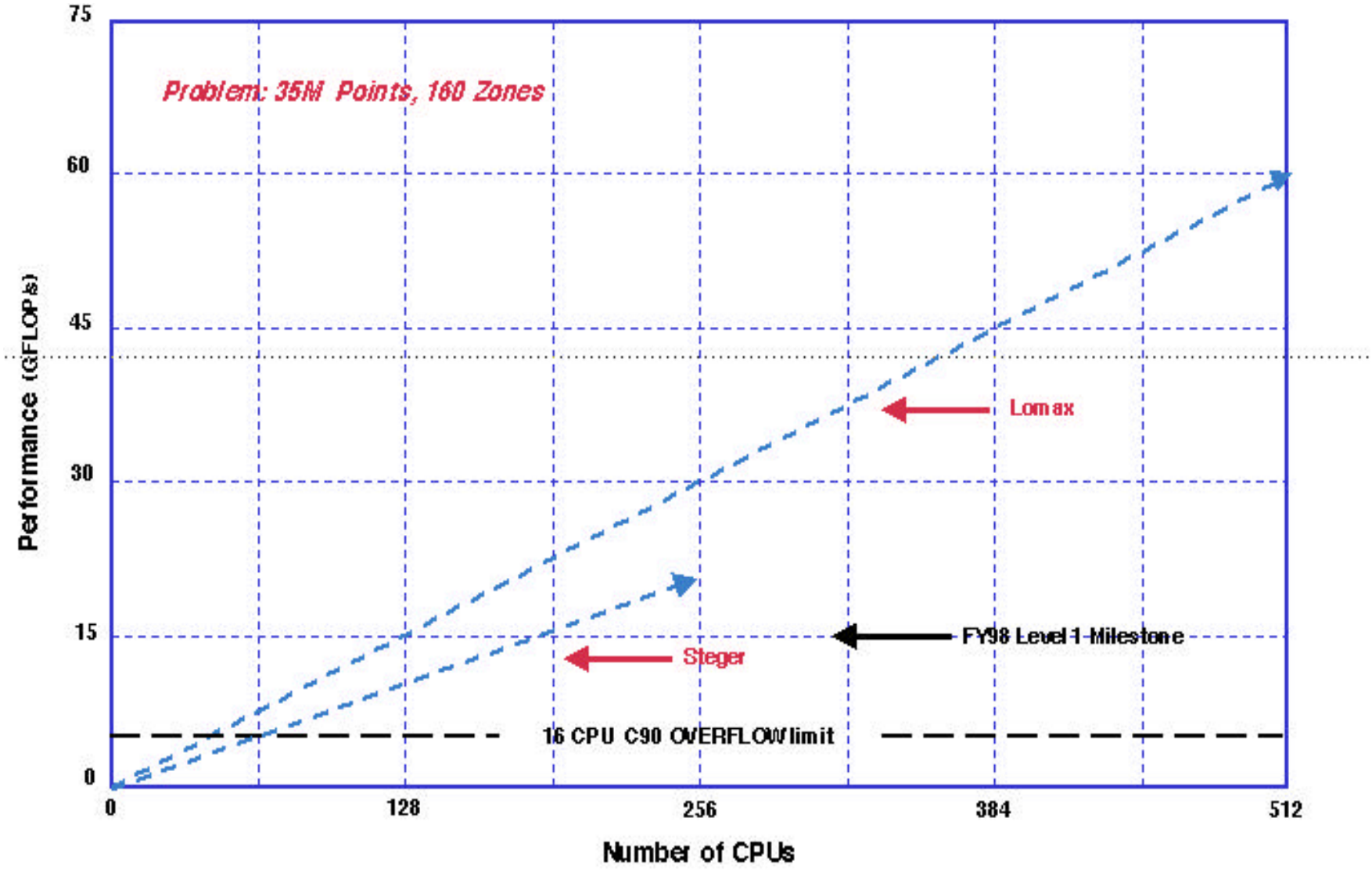
- ***Cost/perf O2K/C90 = 37.7x***
- ***Cost/perf O2K512 to O2K256 = 1.6x***
- ***Performance over 256p = 3x***
- ***Performance over C90/16 = 13x***

NASA Advanced Supercomputing

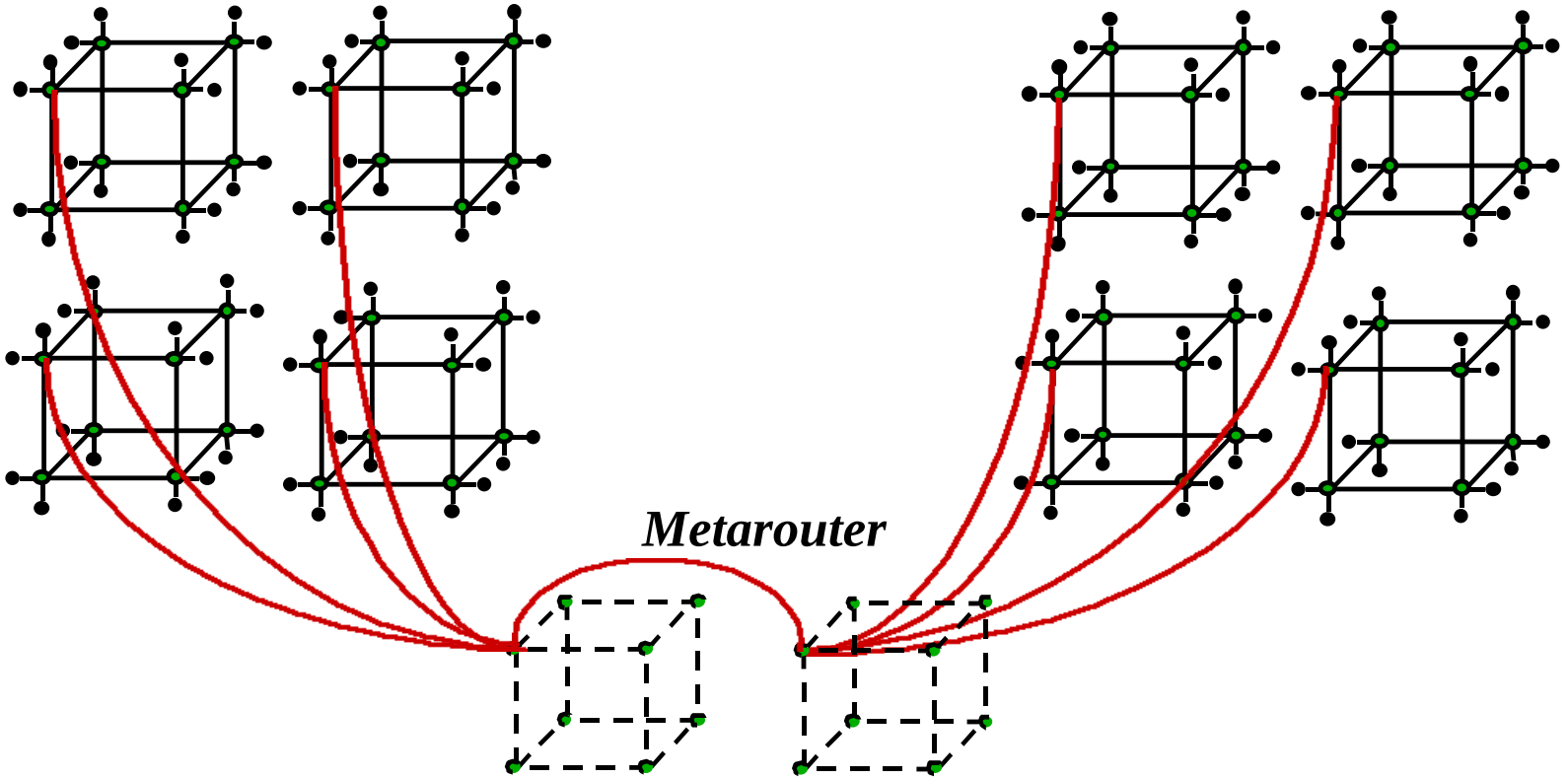
OVERFLOW-MLP Performance vs CPU Count

Steger (250MHz O2K/256p) Lomax (300MHz O2K/512p)

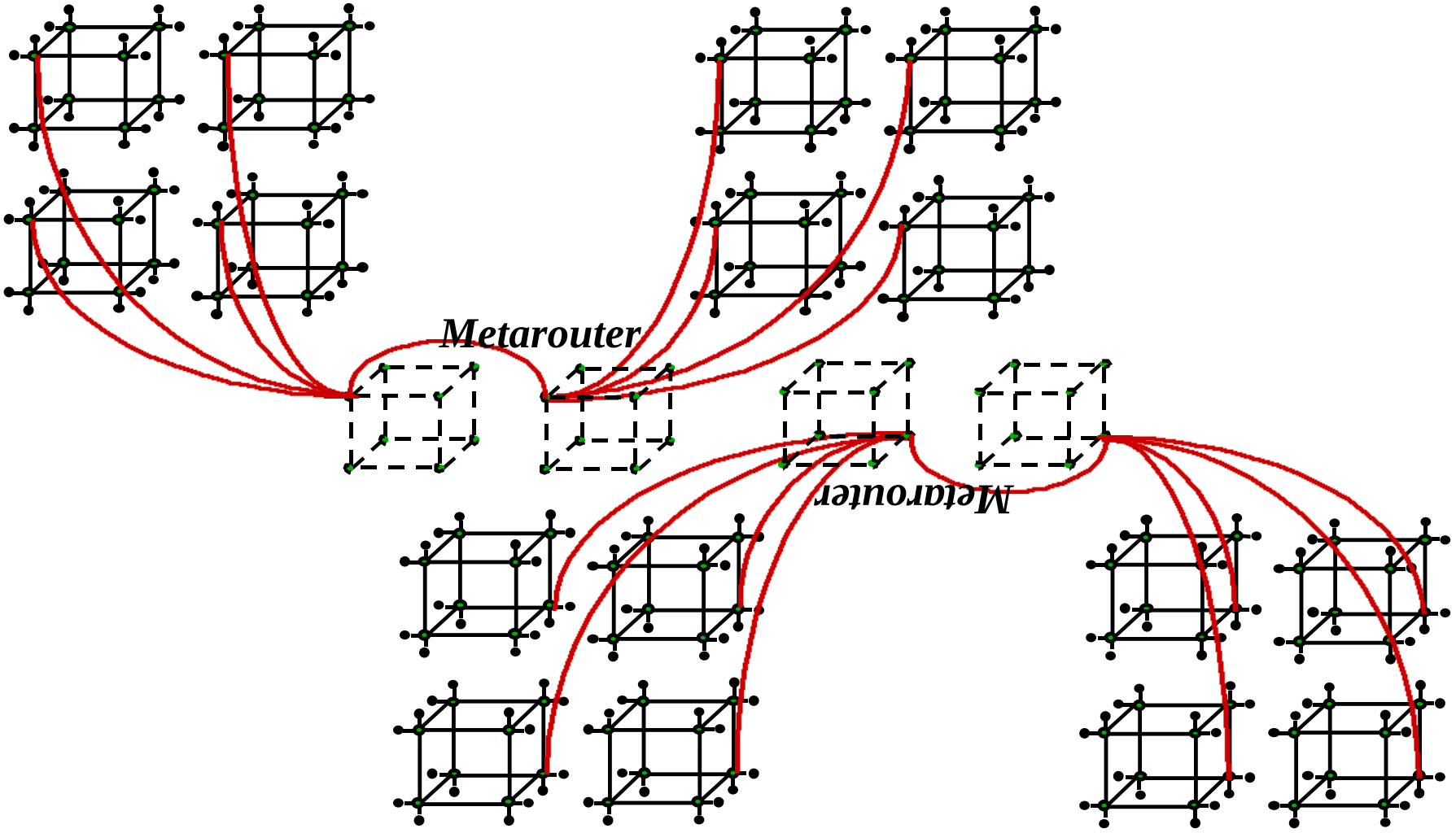
20 Gflops on 256p = 15.6% vs 60 Gflops on 512p =19.5% of peak



256p Single System Image



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

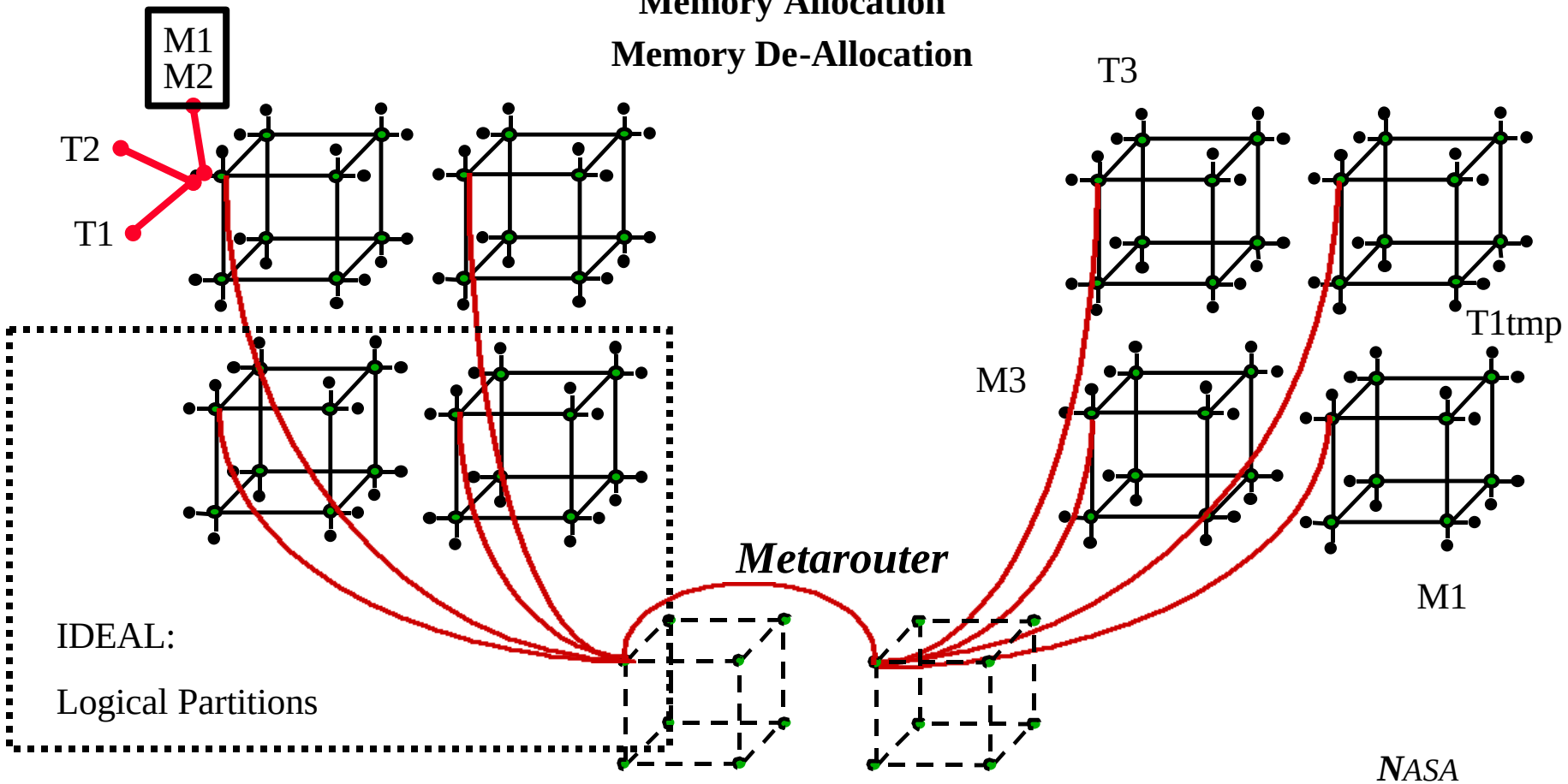


NUMA Aware Operating System

IRIX lacked sufficient control over

Thread Affinity
Thread Location
Memory Allocation
Memory De-Allocation

NUMA effects require that these be dealt with



NASA Advanced Supercomputing

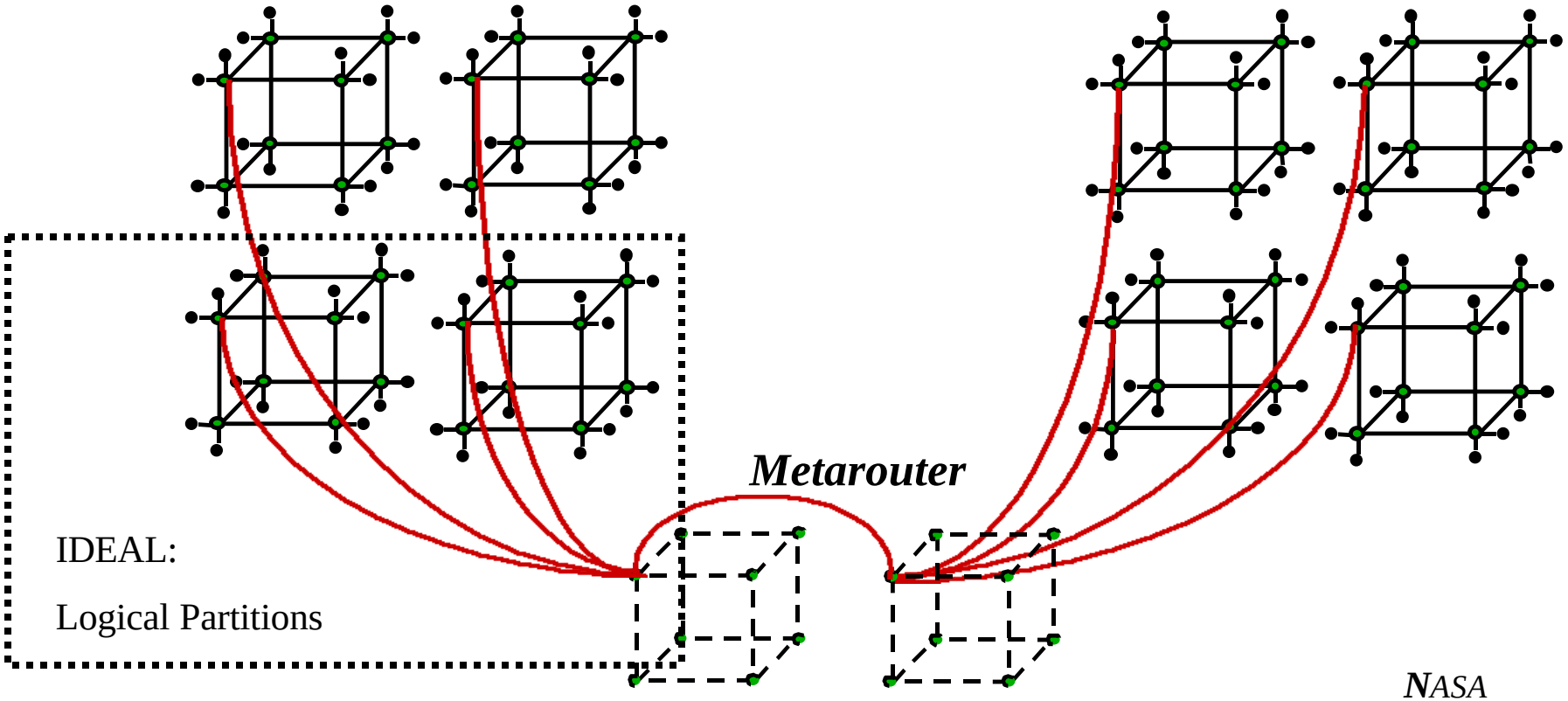
NUMA Aware Operating System

CPU Sets

Memory Limited CPU Sets

Thread Affinity (mustrun)

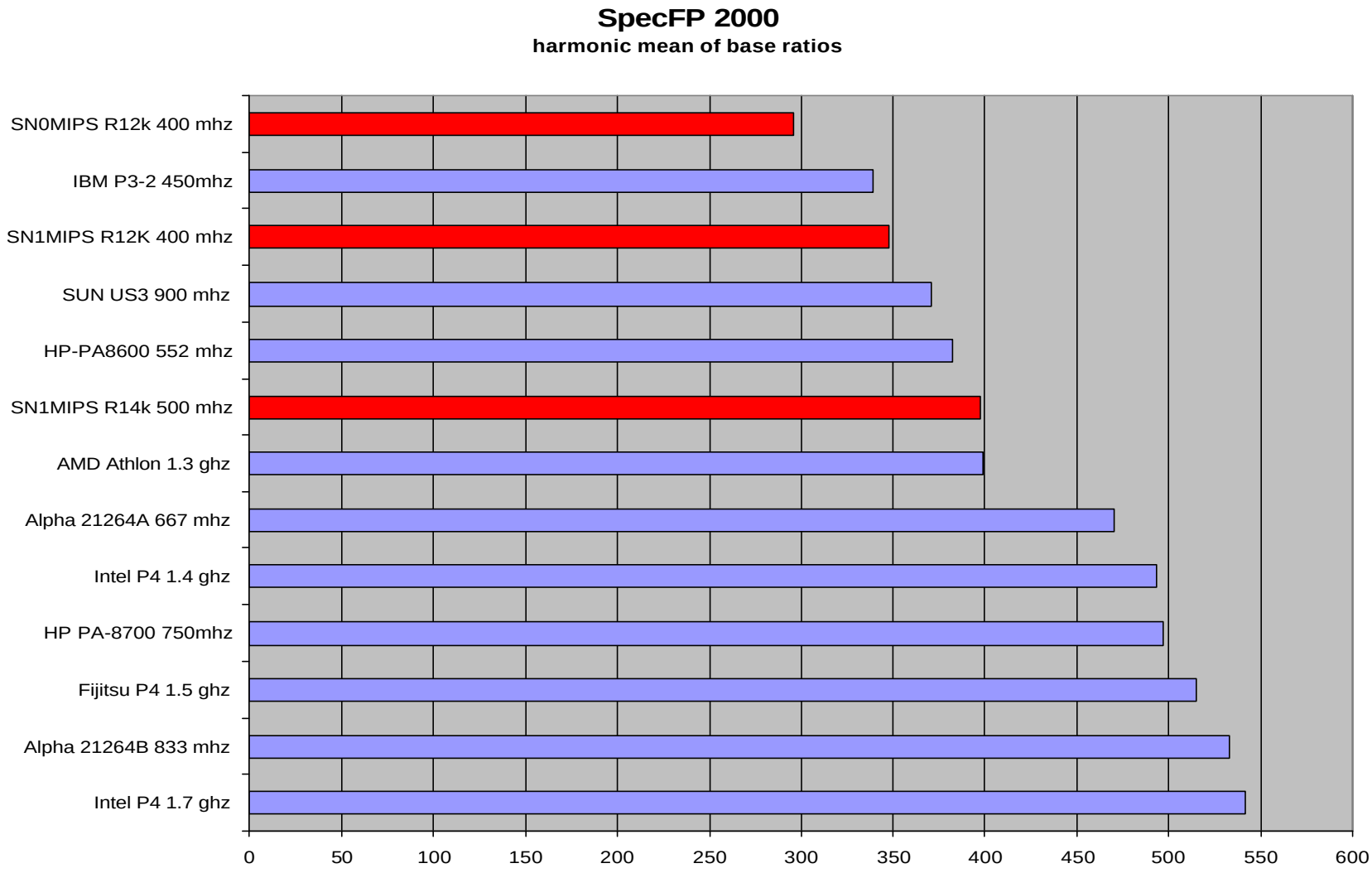
Page Placement (MLD Sets)



NUMA Aware Job Scheduler

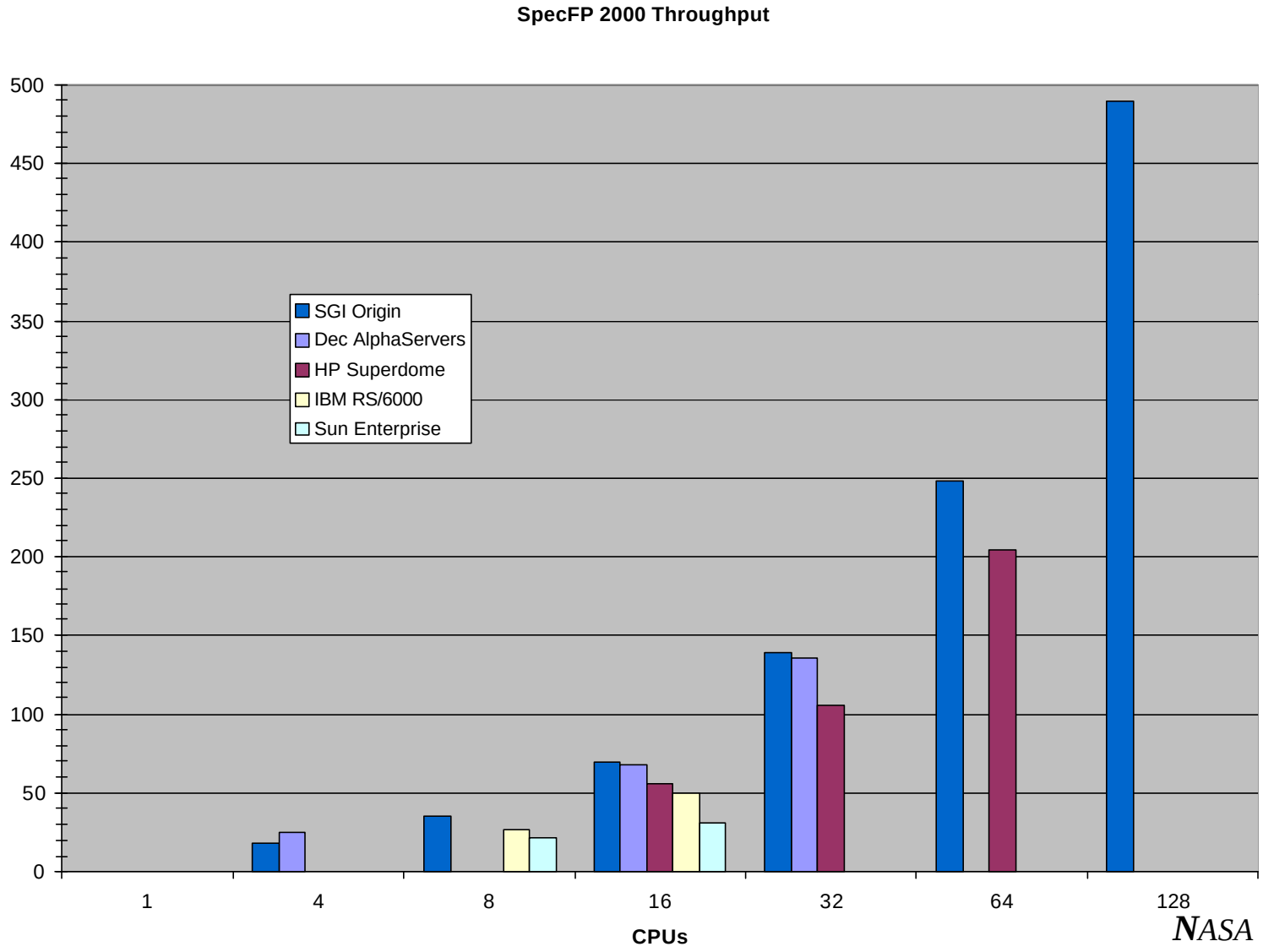
- **Dynamic management of CPUSETs**
  - **CPUSET created of size N at job start**
  - **CPUSET deleted at job termination**

SpecFP 2000
Single CPU



CUG 2001 - Bob
Ciotti@nas.nasa.gov

SpecFP2000 Throughput



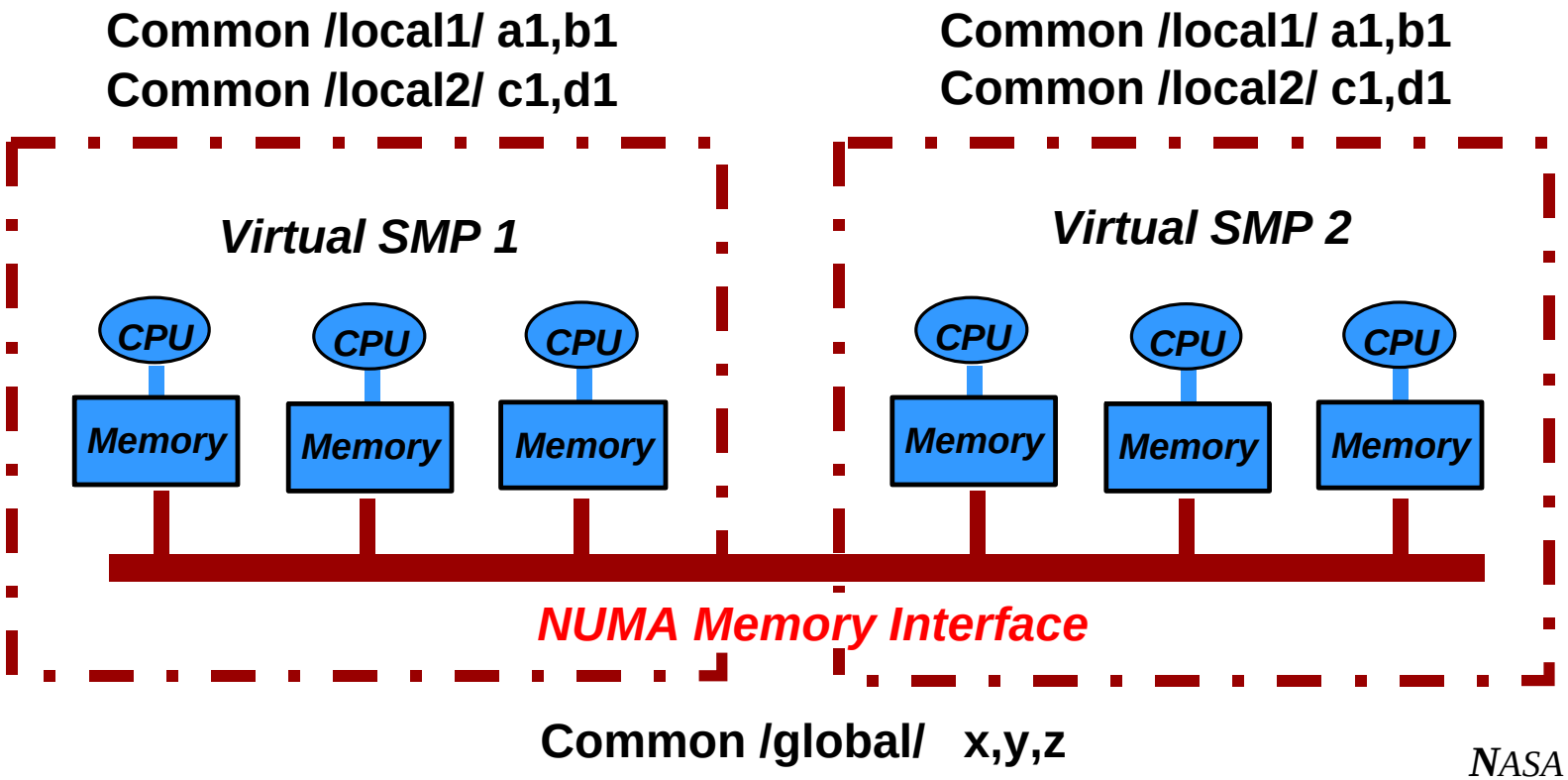
What is Shared Memory MLP?

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.



MLPlib

The MLPlib routines for scalable parallel execution support are:

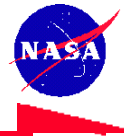
.

- *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(nthread,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 (1-numpro).*
  - *Nthread is an array of ints, indicating the number of OMP threads for each MLP process*
- *Subroutine BARRIER(numpro)*
  - *The BARRIER routine waits until numpro processes have hit the*
  - *barrier, then all drop through.*

Forkit.f

```
c*    Fork the Processes      *
nowpro=1
c-----count offsets to starting cpus
      ival=0
      do n=1,numpro
        write(*,520) n
        istart(n) = ival
        ival=ival+nthread(n)
      enddo
c-----spawn the threads - manual forks
      do n=2,numpro
      nowpid=getpid()
      if(nowpid.eq.master) then
                    ierror=fork()
                    endif
      nowpid=getpid()
      if(nowpid.ne.master) then
                    nowpro=n
                    go to 200
                    endif
      enddo
```

Forkit.f




```
c-----create mp environment
 200 call omp_set_num_threads(nthread(nowpro))

c-----check for pin request
      call getenv('PIN_TO_CPUS',evalue)
      if(evalue.ne.'') then
                  read(evalue,*) idopin
                  endif
      if(evalue.eq.'') then
                  idopin=0
                  endif
      if(idopin.eq.0) return

!$omp parallel
        call mp_assign_to_cpu3(omp_get_thread_num(), istart(nowpro))
!$omp end parallel
```

Mp_assign_to_cpu3 (my_rank, starting_point)

Figure out which physical cpus you have access to

```
pmoctl(PMO_GETNODEMASK_UINT64,&sys_nodemask,sizeof(sys_nodemask))
req.request = CPUSET_QUERY_CPUS;
sysmp(MP_CPUSET, &req)
```

Figure out which physical memory the cpu is attached to

```
sysmp(MP_NUMA_GETCPUNODEMAP,
        (void *)cpu_node_mapping, sizeof(cnodeid_t) * ncpus)
```

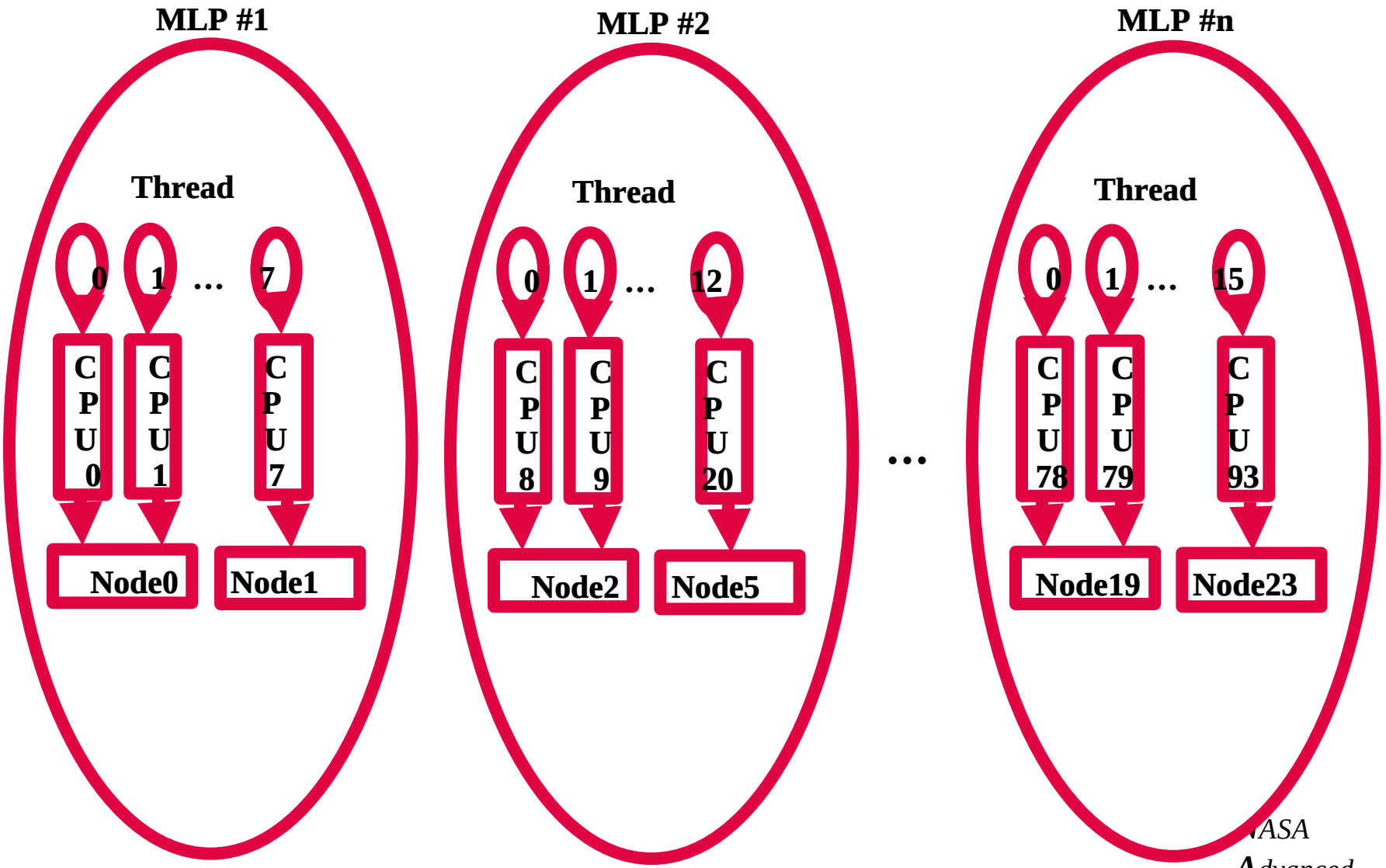
Demand physical pages be allocated from the memory attached to the CPU you will assign to this thread.

```
mld_create(1,1024)
mldset_create(&mld, 1)
mldset_place(mldset, TOPOLOGY_PHYSNODES,
                &affinity_info, 1, RQMODE_MANDATORY);
```

Locks the thread that called this routine to the chosen physical cpu

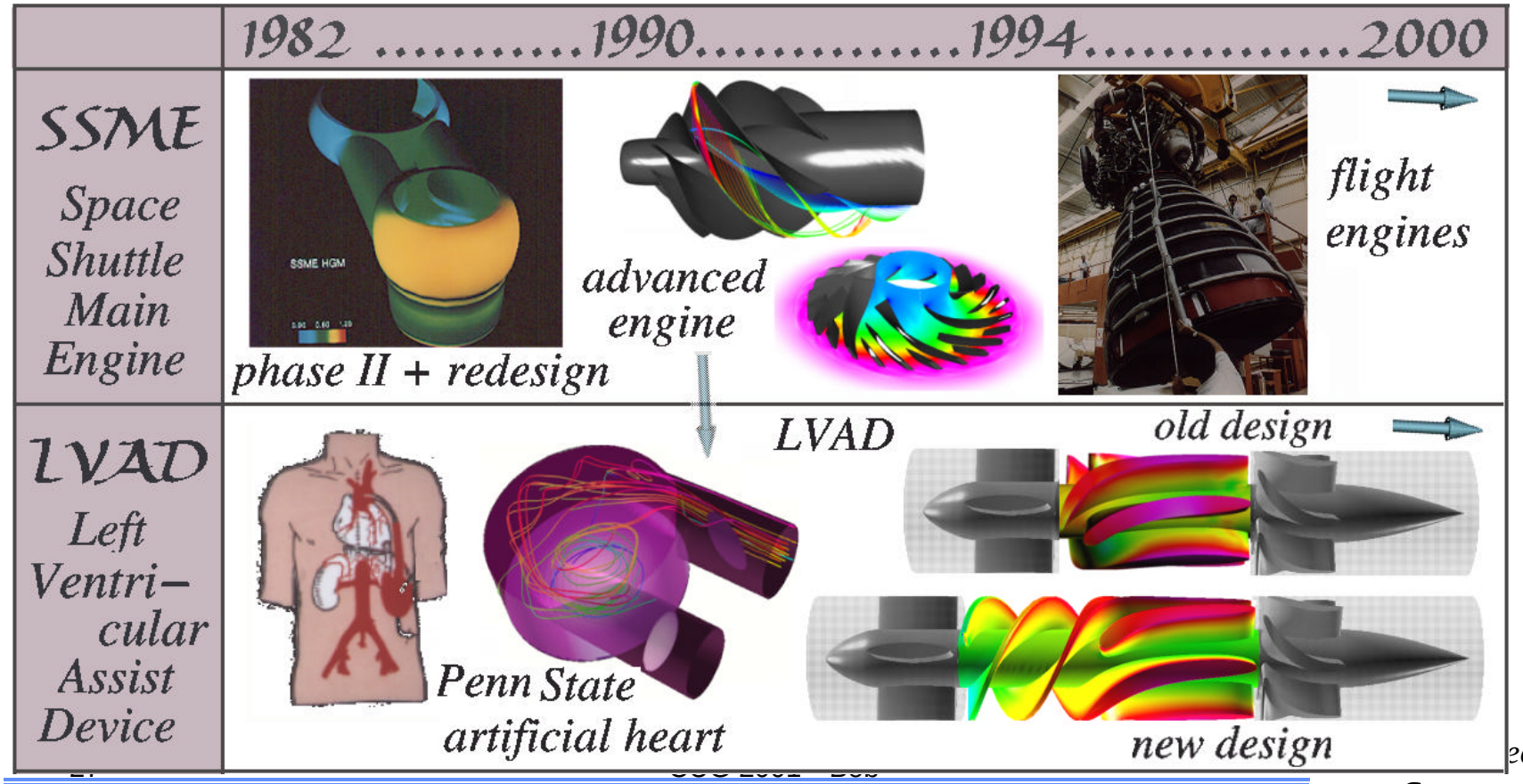
```
sysmp(MP_MUSTRUN, cpulist[my_rank+starting_point])
```

Result - nthread=[8,13,…,16],numpro=N



INS3D Objectives

Enhance incompressible flow simulation capability for developing aerospace vehicle components, especially, unsteady flow phenomena associated with high speed turbo pump.

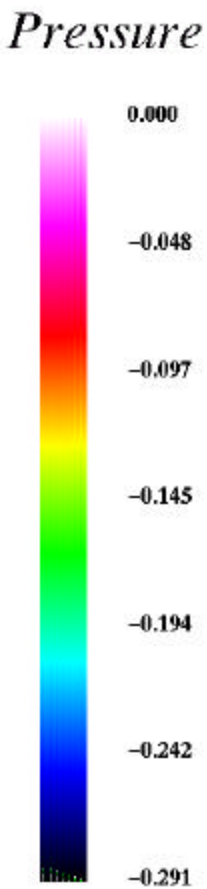


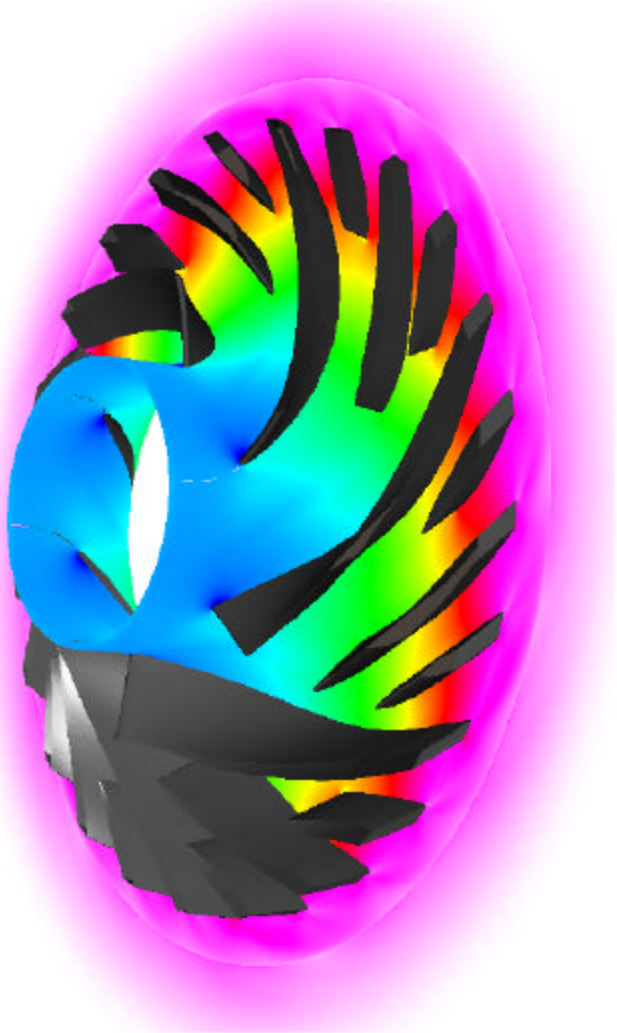
Space Shuttle Main Engine Redesign

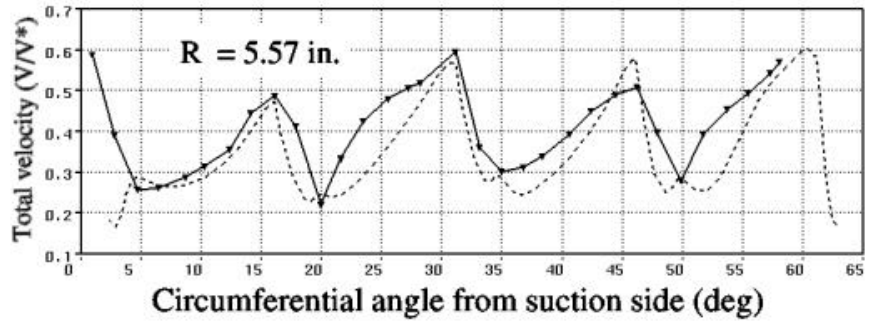
- External Tank filled with approx. 2 Olympic size pools worth of Liquid Hydrogen and Oxygen.
- Main engine turbo pump pushes 1 Olympic size pool every two minutes.
- If water were pumped it would produce a fountain 12 miles high.
- Redesign to reduce approx 1000 lbs of engine weight and increase engine efficiency.
- Will result in a significant payload capacity increase.

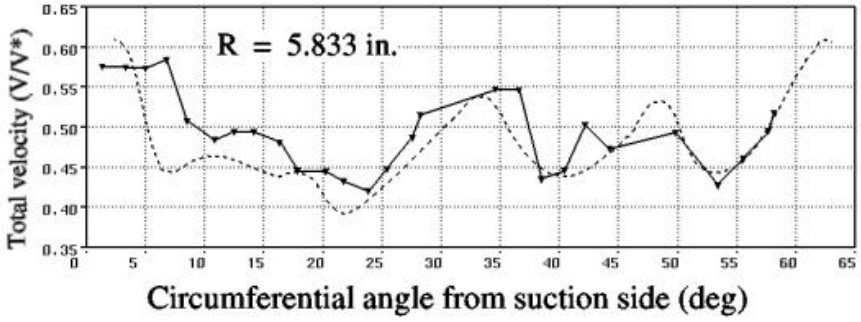
(Fact – First Shuttle Launch
April 12, 1981, 7:00:03 a.m. EST)

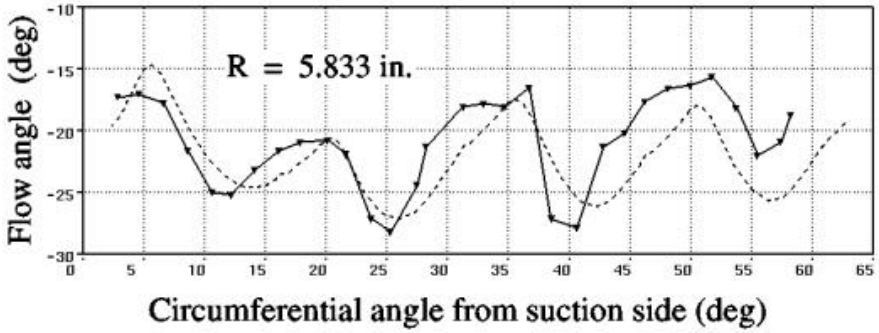
Validation (SSME Impeller)



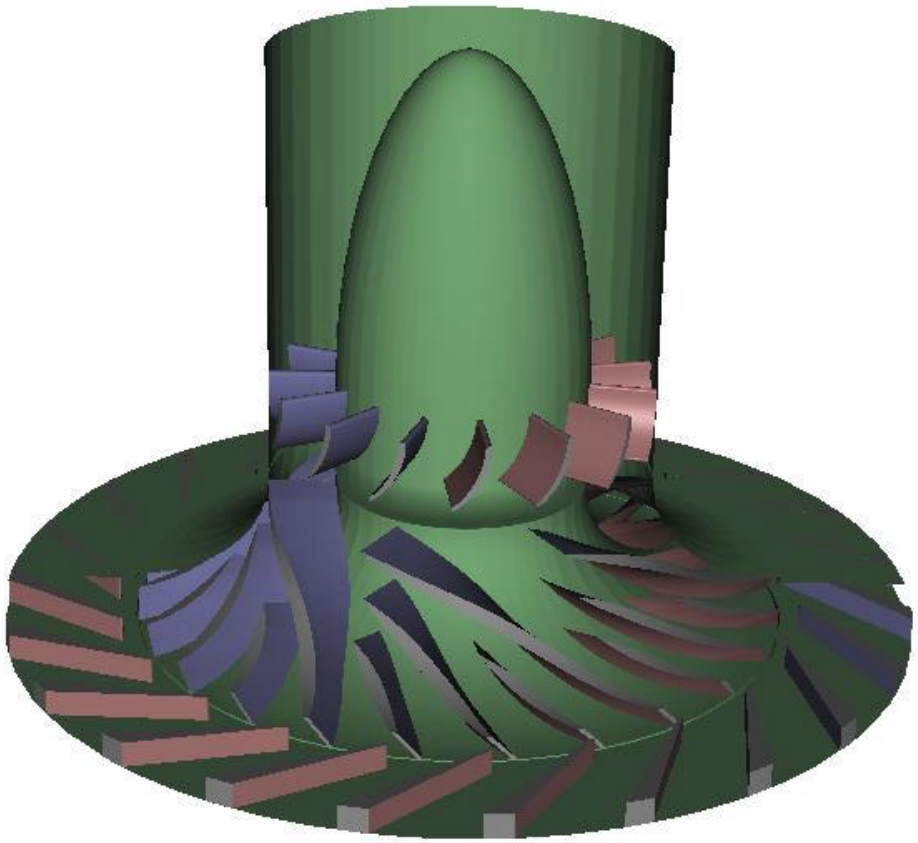








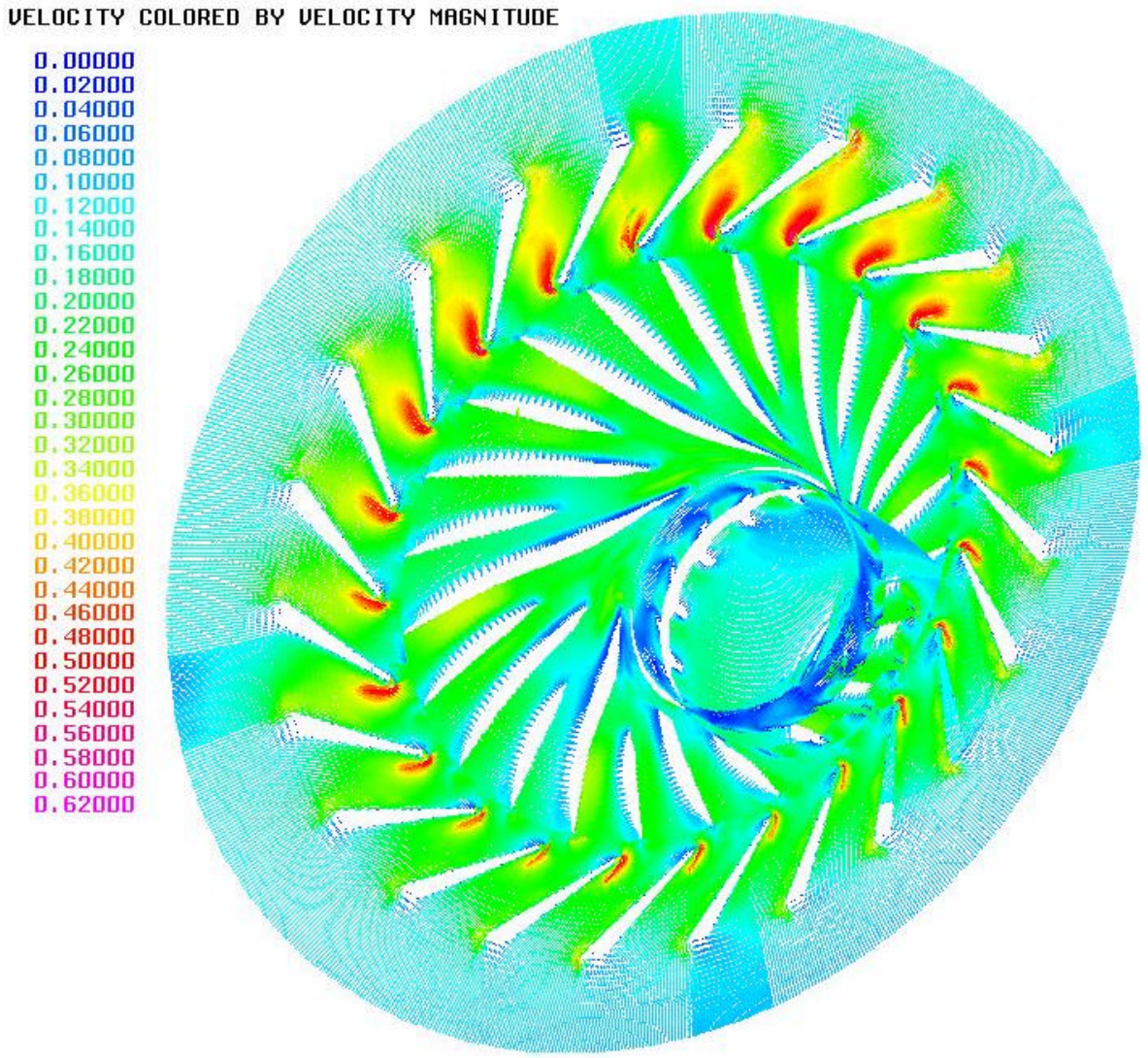
SSME Upgrade - RIG1



- **Full analysis - 3-5 rotations of the pump.**
  - **Takes about a year on a Y-MP**
  - **Currenly about a week on 80 CPUs of Origin3000**
  - **performance goal 1 day**
- **After about 2 weeks on 160 Origin 3000 CPUs, got it to turn 40 degrees!**

SSME-rig1 / Initial start

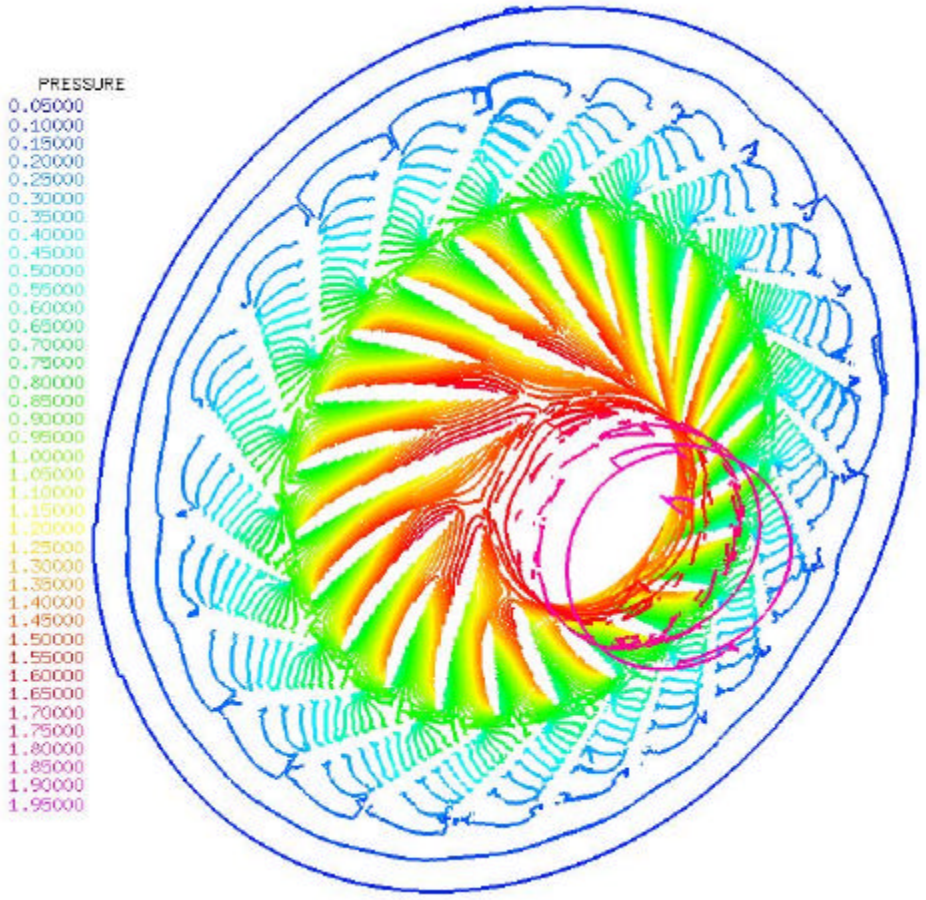
TIME STEP 5



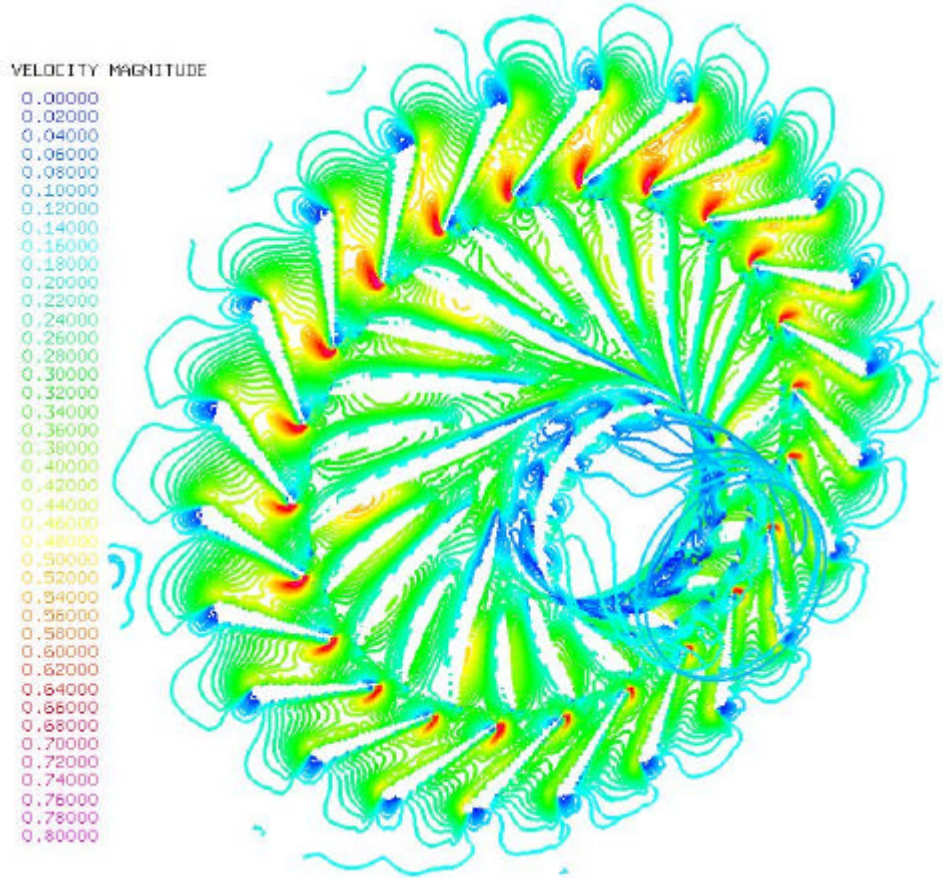
Ciotti@nas.nasa.gov

SSME-rig1 / Initial start

TIME STEP 7 / Impeller rotated 3-degrees at 30% of design speed



PRESSURE

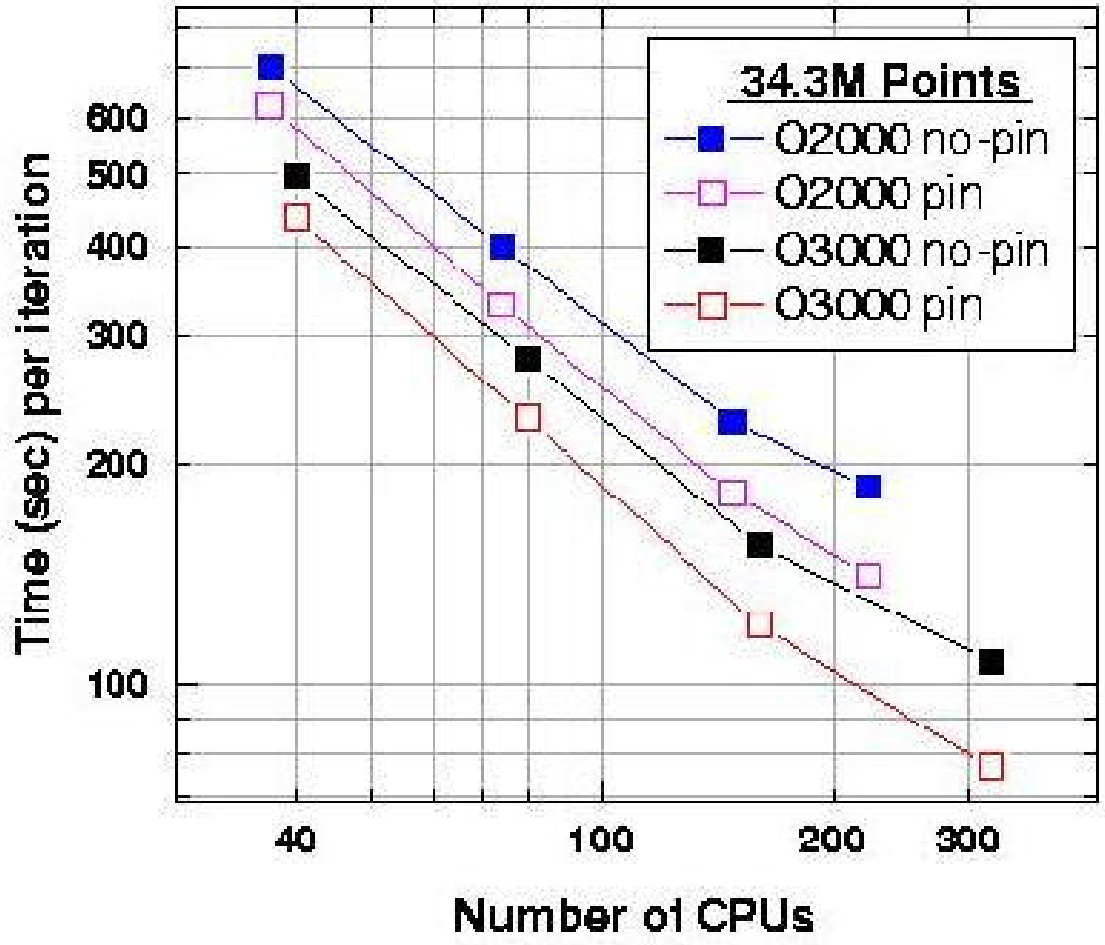


VELOCITY MAGNITUDE

INS3D Parallelization

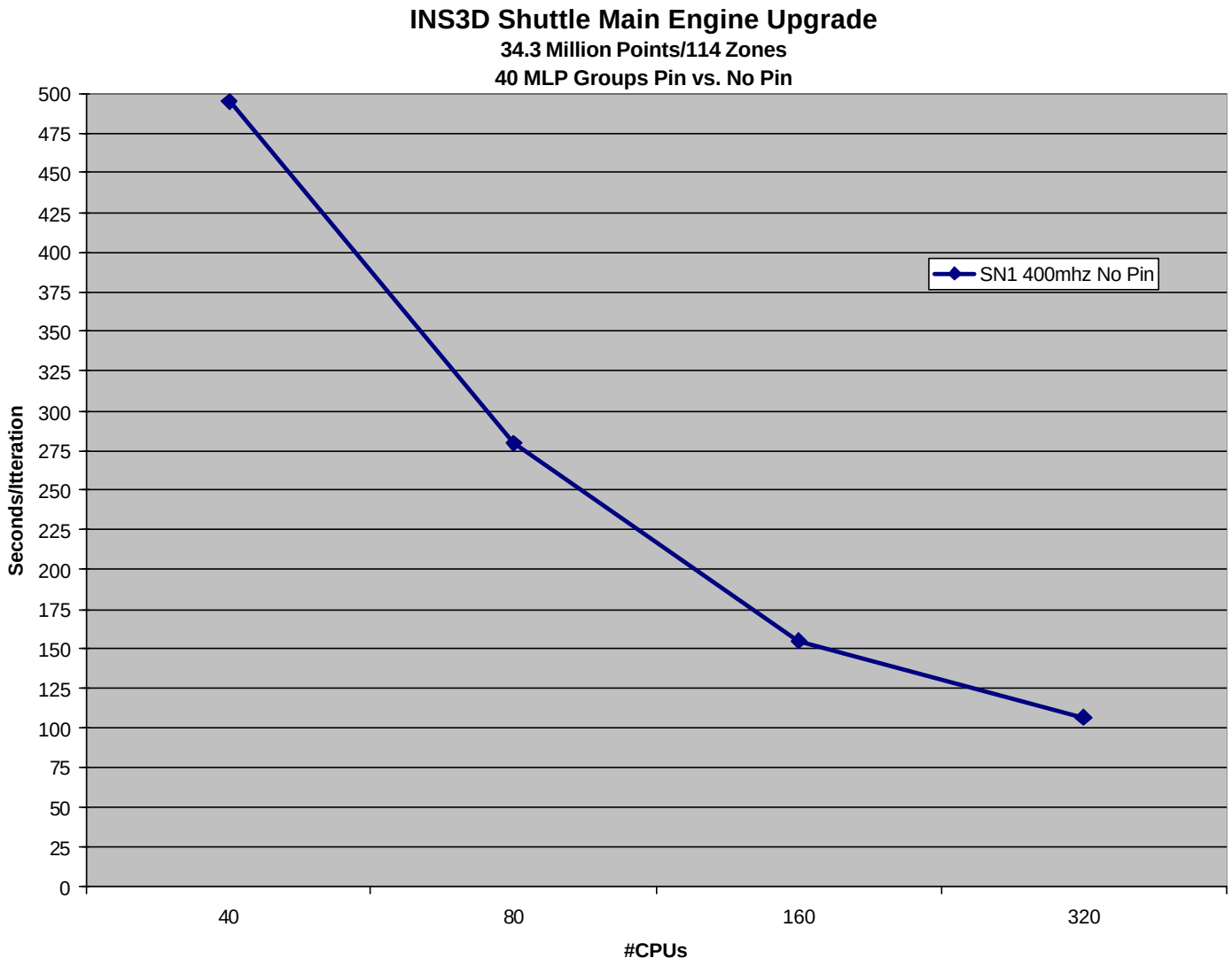
INS3D-MLP / 40 Groups

RLV 2nd Gen Turbo pump
114 Zones / 34.3 M grid points

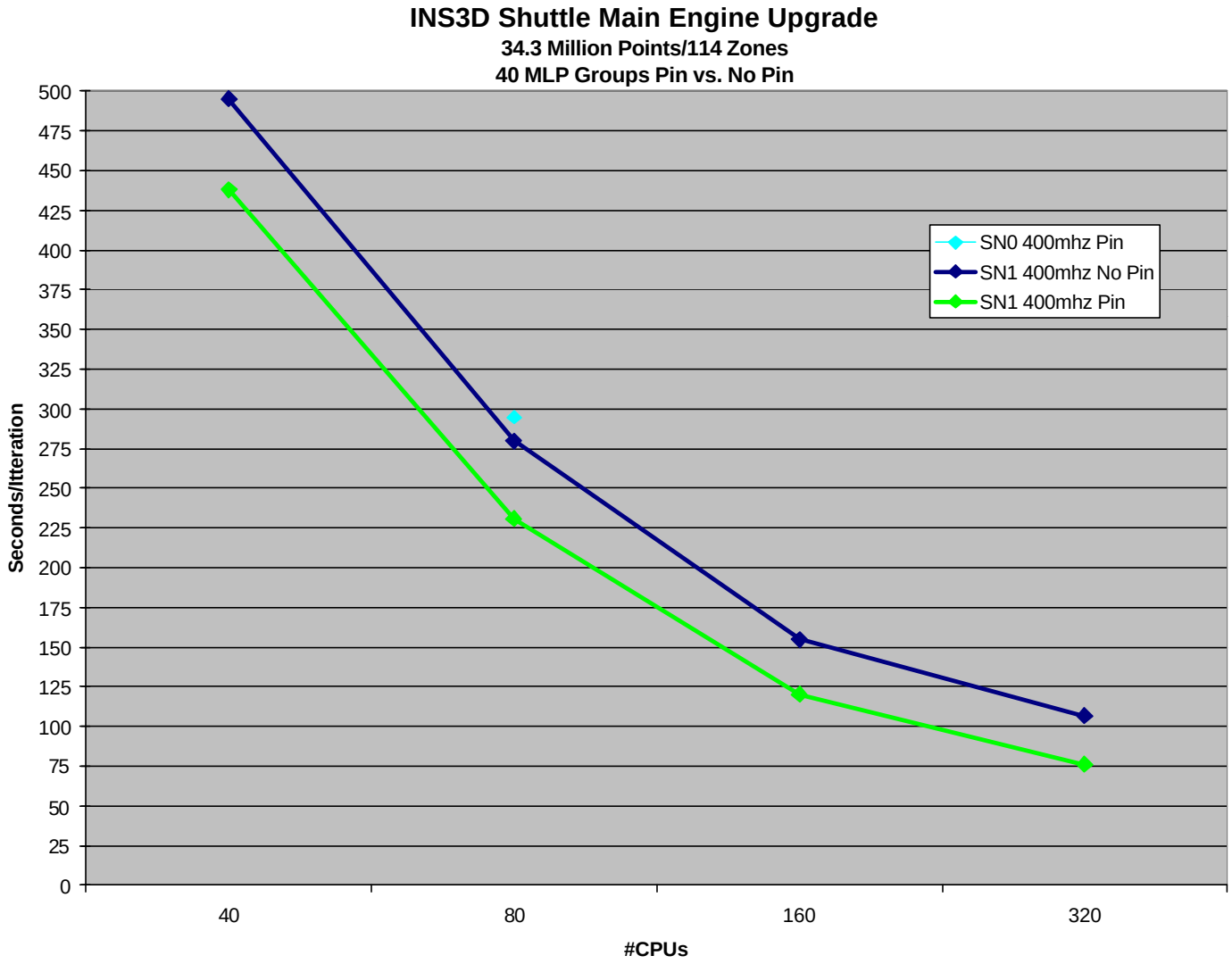


Per processor Mflop is between 60-70. This is a vector code.
Code optimization for cache based platforms is currently underway. Target Mflops is to reach 120 per processor. Increasing number of OpenMP threads is also the main objective for this effort.

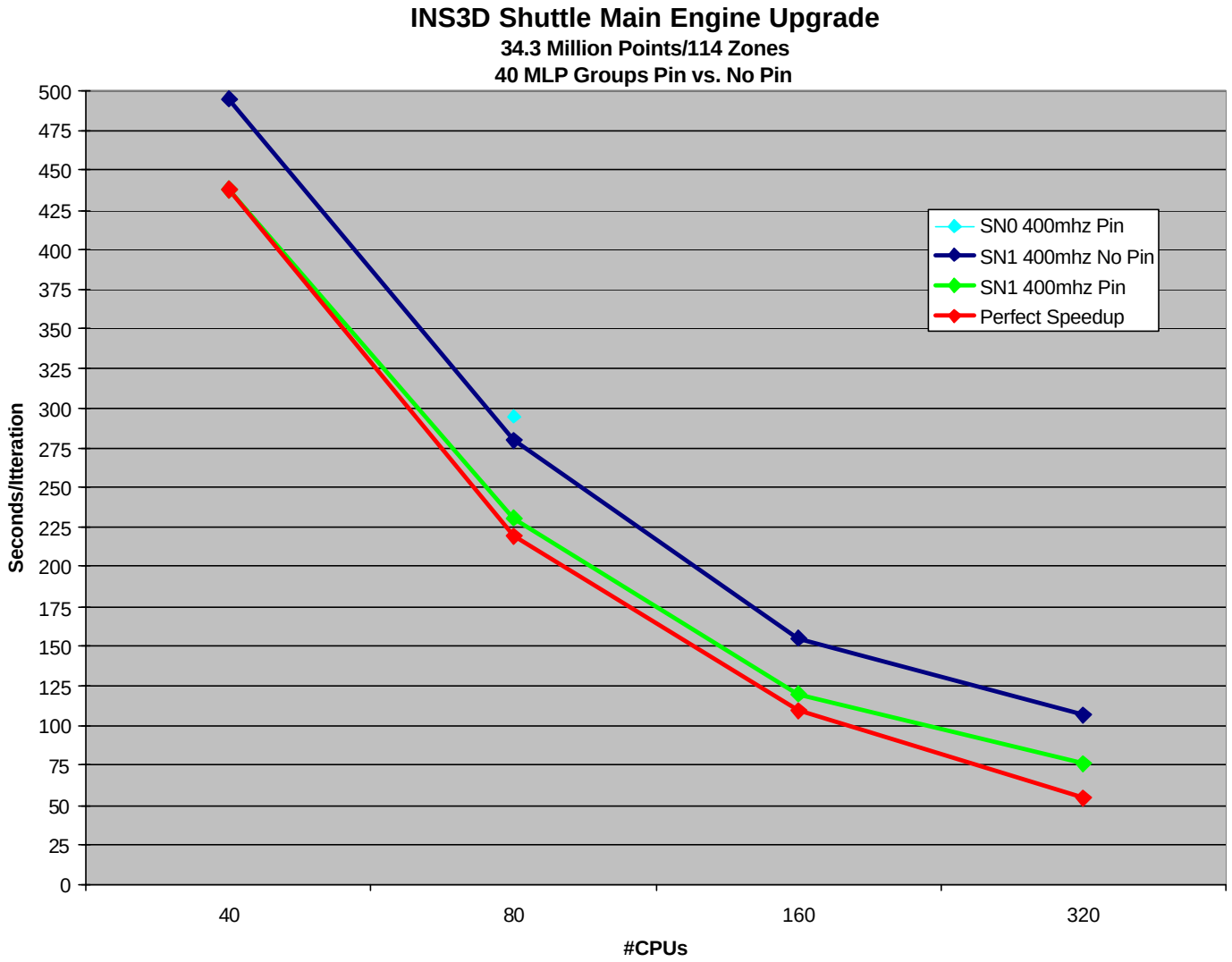
INS3D


INS3D Shuttle Main Engine Upgrade
34.3 Million Points/114 Zones
40 MLP Groups Pin vs. No Pin
Seconds/Itteration
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
40
80
160
320
#CPUs
SN1 400mhz No Pin

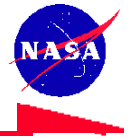
INS3D


INS3D Shuttle Main Engine Upgrade
34.3 Million Points/114 Zones
40 MLP Groups Pin vs. No Pin
Seconds/Itteration
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
40
80
160
320
#CPUs
SN0 400mhz Pin
SN1 400mhz No Pin
SN1 400mhz Pin

INS3D


INS3D Shuttle Main Engine Upgrade
34.3 Million Points/114 Zones
40 MLP Groups Pin vs. No Pin
Seconds/Itteration
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
40
80
160
320
#CPUs
SN0 400mhz Pin
SN1 400mhz No Pin
SN1 400mhz Pin
Perfect Speedup

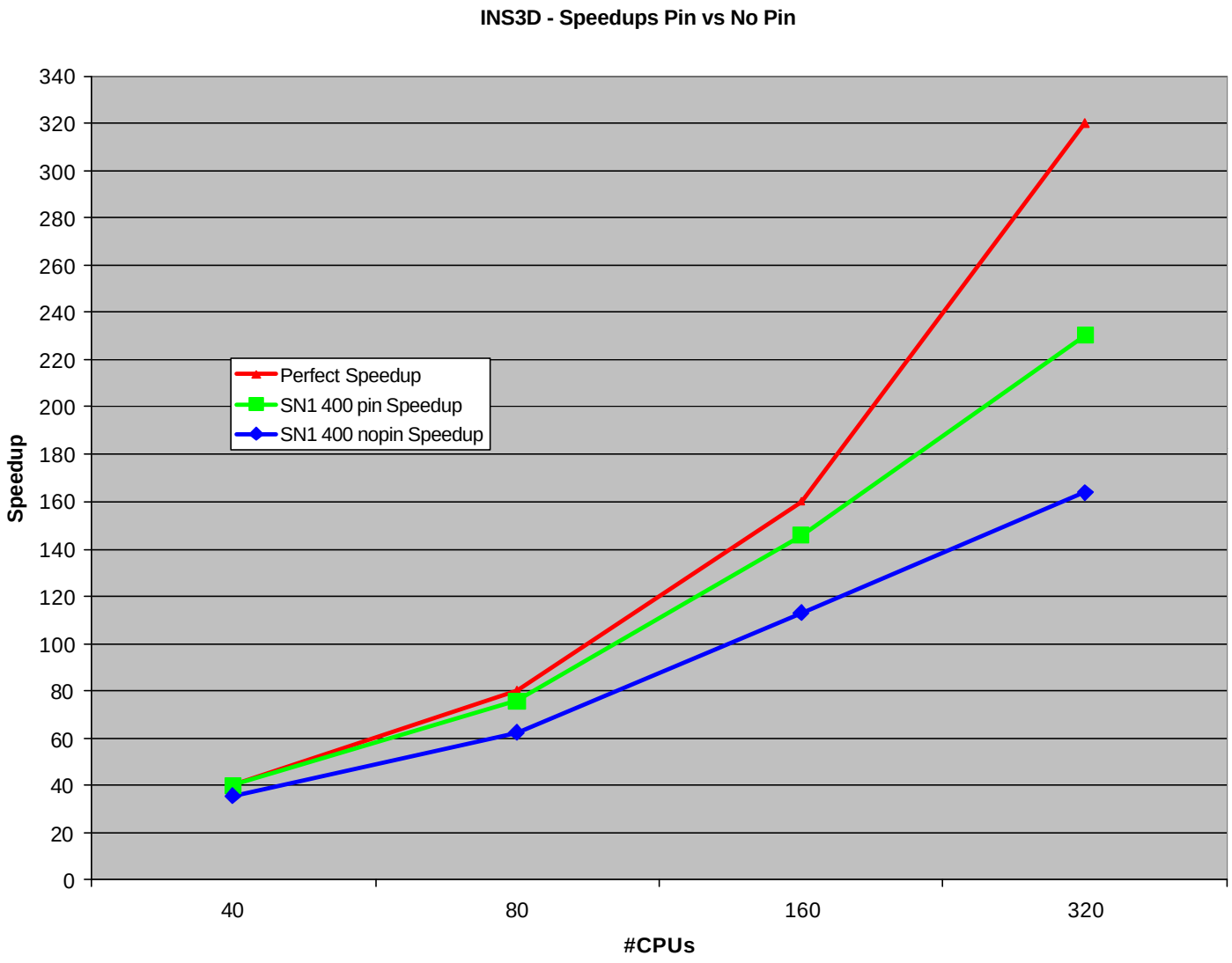
Forkit.f




```
c-----check for pin request
      call getenv('PIN_TO_CPUS',evalue)
      if(evalue.ne.'') then
                  read(evalue,*) idopin
                  endif
      if(evalue.eq.'') then
                  idopin=0
                  endif
      if(idopin.eq.0) return

!$omp parallel
        call mp_assign_to_cpu3(omp_get_thread_num(), istart(nowpro))
!$omp end parallel
```

INS3D

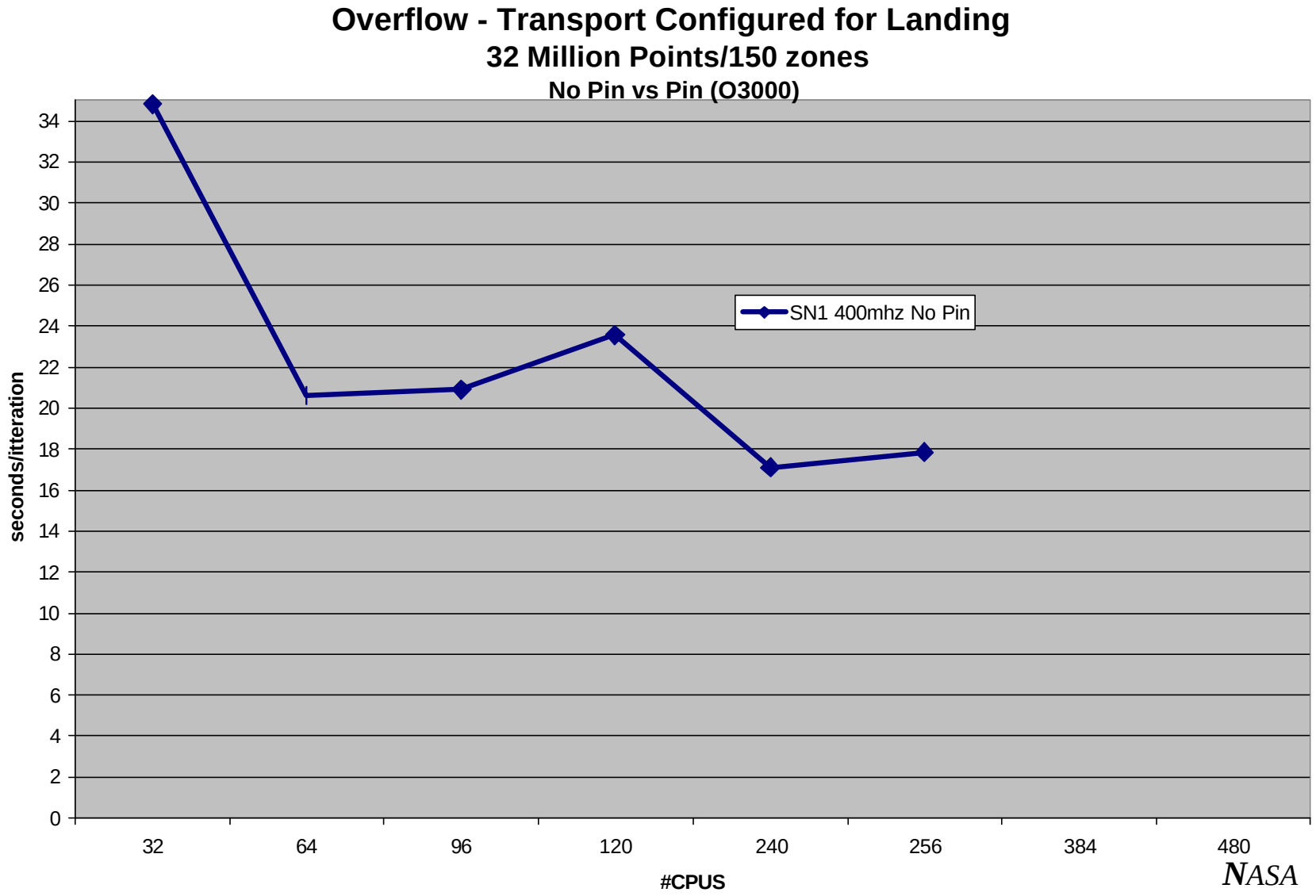

INS3D - Speedups Pin vs No Pin
Speedup
#CPUs
Perfect Speedup
SN1 400 pin Speedup
SN1 400 nopin Speedup
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
40
80
160
320

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**

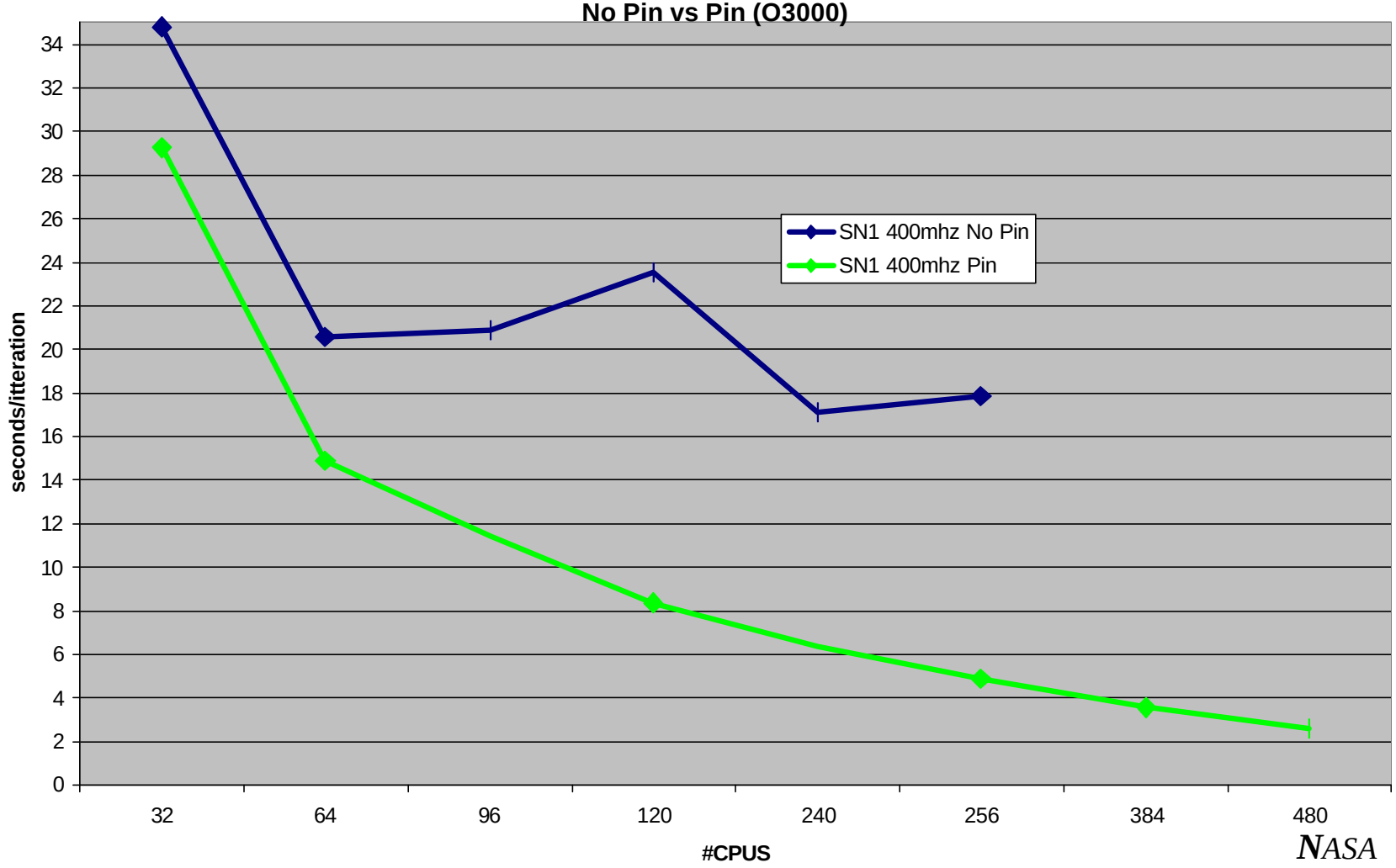
Overflow



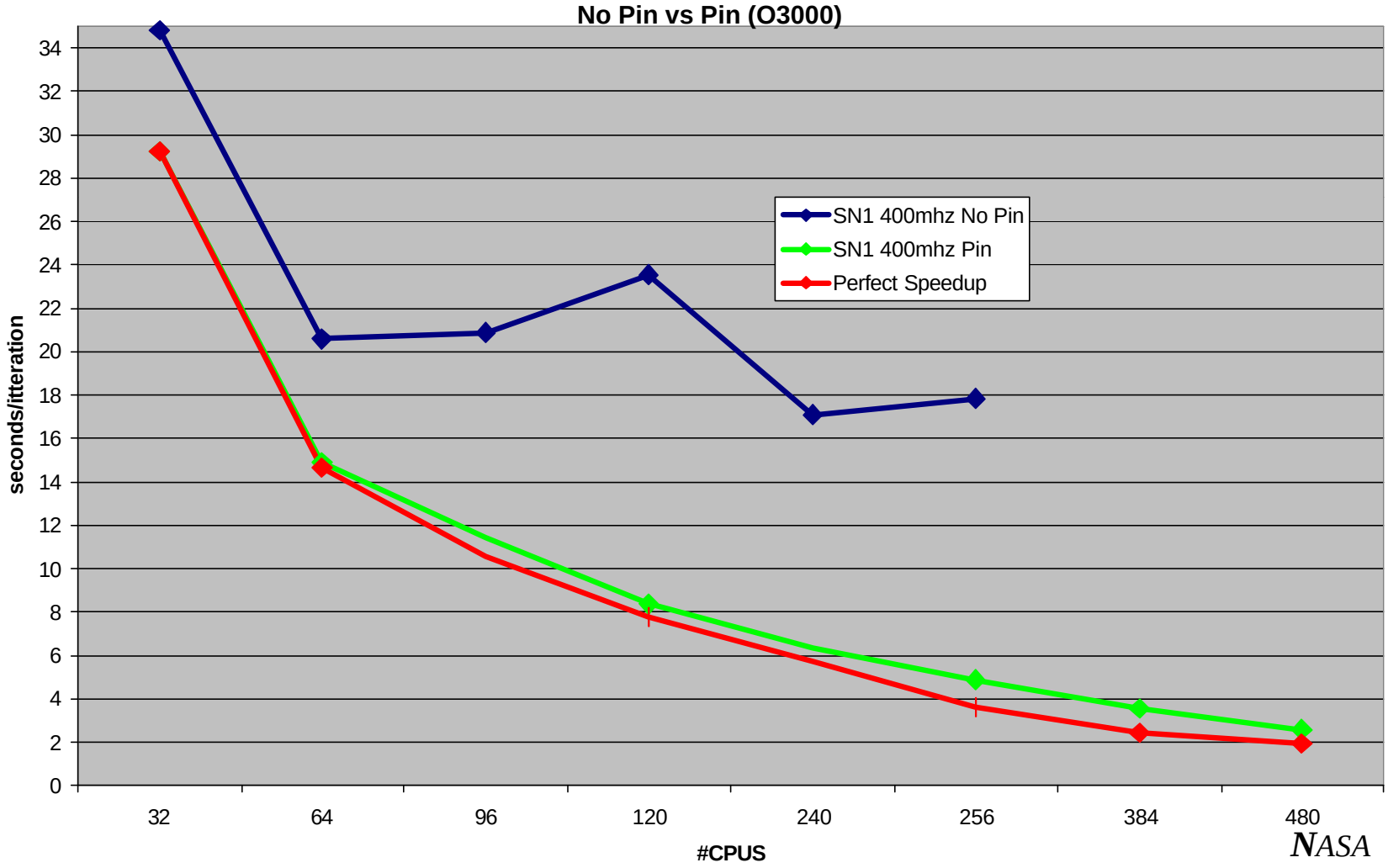
CUG 2001 - Bob
Ciotti@nas.nasa.gov

NASA Advanced Supercomputing

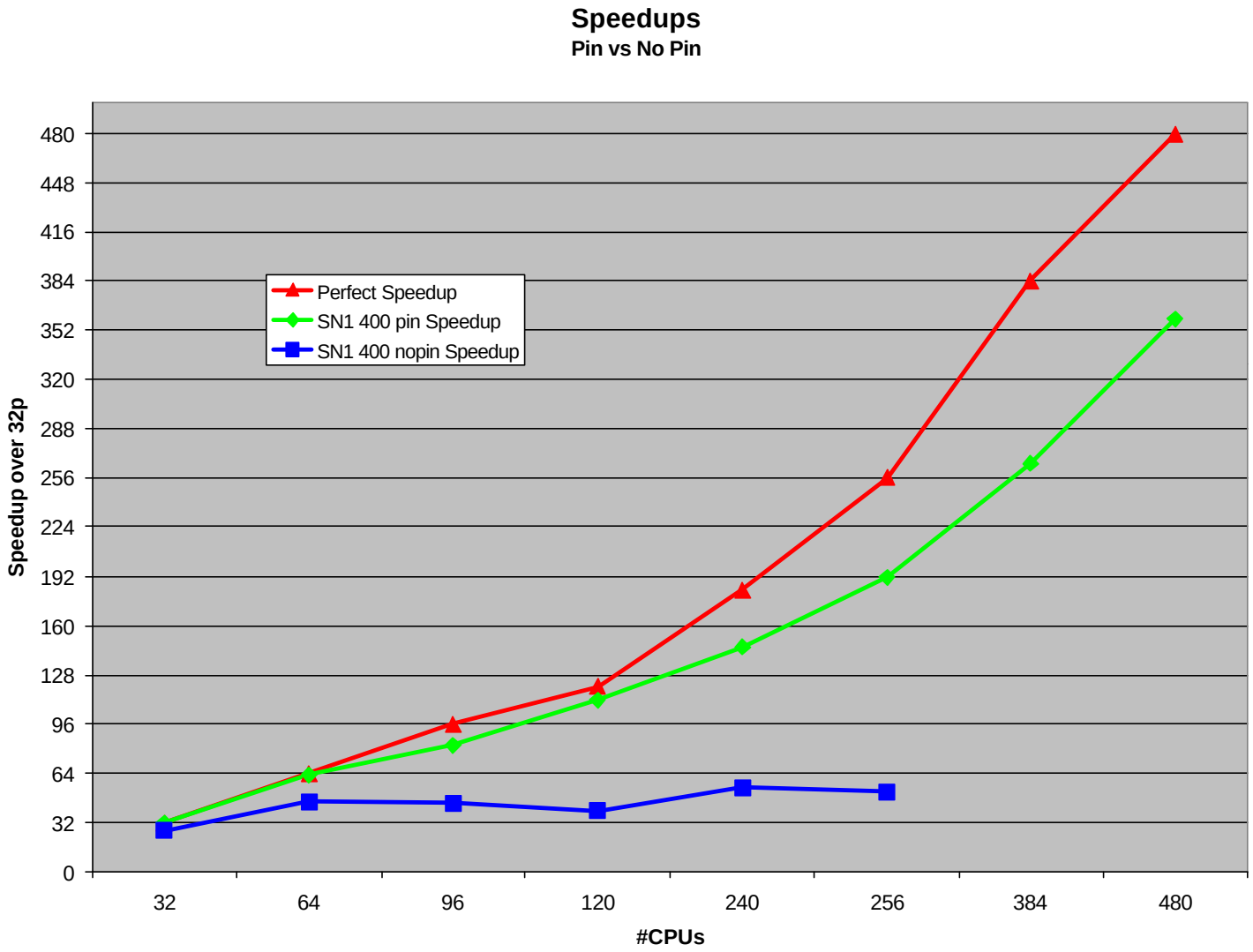
Overflow


Overflow - Transport Configured for Landing
32 Million Points/150 zones
No Pin vs Pin (O3000)
seconds/itteration
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
SN1 400mhz No Pin
SN1 400mhz Pin
32
64
96
120
240
256
384
480
#CPUS

Overflow


Overflow - Transport Configured for Landing
32 Million Points/150 zones
No Pin vs Pin (O3000)
seconds/itteration
#CPUS
SN1 400mhz No Pin
SN1 400mhz Pin
Perfect Speedup
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
32
64
96
120
240
256
384
480

Overflow


Speedups
Pin vs No Pin
Speedup over 32p
480
448
416
384
352
320
288
256
224
192
160
128
96
64
32
0
Perfect Speedup
SN1 400 pin Speedup
SN1 400 nopin Speedup
32
64
96
120
240
256
384
480
#CPUs

1024p Status

- 2 operational 512p 400mhz O3000 system
  - testing and fallout
  - staff and small group of users on systems
- Plan to bring up 1024p in initial topology configuration in June
- Move to higher bandwidth/lower latency topology by end of summer
- Processor speed upgrade around year end +-
- Expectation is to sustain 20% of peak - 240 Gflops

Some Open Issues

- **Page placement doesn't always work as expected**
  - **some (not all) pages allocated via the mld scheme do not end up on the designated node**
- **Some nodes do not de-allocate memory fully when a job exits, forcing future jobs to allocate memory off other nodes.**
- **Mld_create returns a EINVAL with static parameters**
- **Sporadic issues with responsiveness possibly related to I/O activity**
- **CPU X not seen for 20 seconds. Process sleeping at SPL high.**