

# NCEP IBM 2002-2006

George VandenBerghe

I.M. Systems Group

July 2006

# NCEP OVERVIEW

- National Center for Environmental Prediction
- U.S. Primary Numerical Weather Prediction Center.
- History back to 1954.
- IBM Cluster installation since 1999.
- Computing history of the site was described in SCICOMP VI presentation
- <http://www.spscicomp.org/ScicomP6/Presentations/VandenBerghe>

# PREVIOUSLY DISCUSSED NCEP EXPERIENCE

- Site has always had most advanced supercomputers.
- Paradigm went from Pipelines to Vector to SMP to Distributed Memory Message Passing.
- All paradigms map well to NWP and Climate Analysis.
- All platforms including P3 and P4 clusters have been successful supporting these problems.

# NCEP COMPUTING

- There have been ~~28~~ 31 doublings in compute capacity since 1950.
- The crossed out figure was from the Summer 2002 Spscicomp presentation.
- Long term site is following “Moore’s Trend”
- Slight additional increment from processor count increments (will this soon change??)

- Improvement breaks down to  $2^{20}$  improvement in processor speed and  $2^{11}$  increase in processor count.
- On a flat interconnect with current metrics, NWP forecast algorithms scale linearly to our processor count. (We can use more processors).
- Current NCEP decompositions compromise scalability (but not very much) for cpu efficiency or implementation simplicity.
- There's a LOT of opportunity to use a 10x-100x increase in processor count.

# Current Platform

- Two P655 clusters each with....
- 1248 batch P4+ processors in 160 compute nodes. 16GB/node.
- Federation Switch 4GB/Second/node.
- Apps do 500-1000mflops/CPU.
- 40TB of GPFS on one cluster (primary ops)
- 80TB on second cluster (primary development).
- 1.5PB HPSS archive. (10000 STK 9940B tapes)

# Primary APPS

- Global Spectral Model (GFS)
- Regional Grid Point Model (North America and surrounding oceans). (NAM → WRF)
- GFDL Hurricane Model.
- Ensembles of 1. and 2.
- Very short range update forecast (RUC).
- Air Quality Model (AQM).
- Assimilation systems to support above.
- Development work to implement next generation of all of above.

# Recent Platforms

- 2002, Two WHII clusters with TB switch, 256 4 cpu nodes.
- 2003Q1 Two P690 clusters 160 4 cpu nodes apiece. Colony Switch
- 2004Q4 Two P655 clusters 160 8 cpu P4+ nodes apiece Federation switch (current system)
- 2006Q3 (upgrade to ~160 16 cpu P5 nodes).

# P690 Platform (Frost/Snow)

- 1.3GHZ P4 cpus 4/node Colony switch.
- LL preemption introduced with this machine.

# LL Preemption Experience

- Desire was to get High Priority production work to displace Low Priority Development work, then allow Low Priority to resume after the production work.
- The high priority displacement worked (with caveats)
- Getting the low priority restarted has proven to be tough!

# LL PREEMPTION preempt

- If jobs use little memory, preemption is fast.
- Jobs that use a lot of memory must page out their address space as High Priority needs it. This is VERY slow.
- When we detect such paging, we kill the low priority in RSCT monitor scripts. Prod must go through consistently!
- After a learning curve this has been rare.

# LL PREEMPTION PREEMPT

- Some problems with parallel jobs running serial steps.
- These use a cpu and do not take sigsuspend.
- Prod jobs that need all cpus (MPI jobs generally) are grossly slowed by these!
- This is rare but frequent enough to cause problems.

# LL PREEMPTION RESUME

- Jobs stay preempted “forever”
- E state is dreaded by developers. Many cancel and try again.
- Problem is that small jobs flow in around the prod jobs, land on the nodes used by preempted jobs and prevent the preempted jobs from restarting.
- Worst with combination of mixed dev workload and small prod jobs.

# LL Preemption Image Accumulation.

- Large Dev gets preempted by small prod.
- New large dev starts on the first job's nodes.
- New large Dev gets preempted by small prod.
- Newer large dev job starts on remaining nodes.
- Repeat until page space exhaustion and crash.

# P655 (Blue/White)

- 2x1280 cpus, 160 8 cpu batch nodes  
16GB memory 1.7GhZ P4+ processors.
- Federation Switch ( huge improvement!)
- 21 and 23TB of GPFS augmented with 70  
and 20 TB 2005.
- Overall a big boost to capability (3.2x  
P690 clusters)

# New Issues

- Increased memory size makes paging more visible and less tolerable.
- Larger memory and cpu count allow wider SMP mix. Good but potential for cpu oversubscription increased.
- Overall very good machine!

# Paging Problems

- Memory size has increased 32x since 1999.
- Page space speed has increased 4x (that's generous!)
- Paging memory to disk is a slower and slower operation.
- 1980 1mb/sec disk 1mb memory 1 sec to page.
- 2004 40mb/sec disk 16GB memory. 400 sec to page.
- We kill heavy pagers.

# CPU Oversubscription

- If product of taskcount and threads >cpucount, cpus will have to timeslice.
- The GROSSLY slows MPI jobs and can't be tolerated!
- On this system we have no control over it (but on new pending system, consumable cpus addresses and mostly corrects it)

# Switch

- WH and WHII used 250(160)mb/sec Tb switch
- P690 used 500(340)mb/sec “Colony” switch
- P655 used 5(4.2)GB/sec “Federation” switch.
- Federation is a huge huge step forward!
- 16x bandwidth 4x latency improvement
- watch latency:bandwidth ratio.

# New Problem. I/O

- NCEP apps have not been I/O bound for 20+ years
- Isolated slowdowns 2003 and 2004 when a GPFS server was down.
- Slowdowns now sometimes occur with GPFS infrastructure healthy.
- IOSTAT on the GPFS servers show the supporting disk is saturated.

# I/O

- Originally had 21TB of GPFS 1gb/sec.. not enough space (or bandwidth) for users + prod.
- Added 70TB split into 11TB of scratch (/ptmp) and 59TB of permanent quota controlled space.
- /ptmp became a 400mb/sec throughput bottleneck.
- Gridlock at /ptmp spread to /nwprod due to bottlenecks in GPFS servers.
- Permanent bandwidth (1.5GB/sec) remains underused.
- Will go to one big GPFS with new filesets on next system.
- Exercise above exposed I/O issues and will overall be beneficial.

# I/O control

- Tools to detect who does I/O and how much, where are more primitive than cpu and memory tools.
- I/O bandwidth is not as easy to partition between workloads as CPU.
- This remains an issue.

# MASS STORE

- Old Tivoli TSM was a weak point 2002.
- New IBM HPSS much more solid.
- HPSS sustains 140-160mb/sec into/out of the HPSS mover cloud. 50mb/sec single stream.
- About 1PB of data image. Most dual copy thus 2PB of tapes. (of course it's full!)
- Reliability of newer 9940B tapes suggests dual copy may not have been necessary.

# Disk Bandwidth Trends

- Early NWP was few deterministic models.
- These produce a single set of forecasts.
- Image sizes do not grow as fast as cpu because of CFL constraints.
- People haven't paid a lot of attention to I/O.

# Ensemble Problem

- Recent years have seen increases in NUMBER of forecasts rather than size of the individual ones.
- In this environment disk space and bandwidth scale with CPU capability.
- Technology has not scaled this way

# Comparison

- 1982 CDC CYBER 205
- 100mflop 3mb/sec to disk 15mb/sec agg
- 2004 P655 cluster
- 1Tflop 200mb/sec to disk 1500mb/sec agg.
- Cpu increased 10000x disk rate 100x.
- Note we're talking rate, not size which has kept up.

# Mass Store Comparison

- 1982, primitive network from C205 to front end tapes 200kb/sec (10MB/sec online tapes)
- 2006 200mb/sec to HPSS.
- Transfer rate to tape farm increased 1000x compared with 10000x cpu increase.
- If we look at online tapes that ratio is a lot higher 10000:20

# I/O issue summary

- We're nearing bandwidth saturation on both disk and tape.
- On disk it will cause substantial runtime variance.
- On tape it will result in sluggishness of high level user operation ( "get 1/10 your disk to tape" for example)
- 1987 1GB 200kb/sec 5000 sec
- 2006 12TB 200mb/sec 60000 sec

# Coming Soon

- P5 cluster 2000 P5 processors (x2)
- ~150 16way 32GB nodes. 150TB GPFS (heavily optimized for bandwidth)
- Consumable Cpu and Memory for first time at NCEP. Very promising in early tests.
- Consumable memory effectively prevents paging.
- A few loopholes but consumable cpus effectively prevent cpu oversubscription by threaded jobs.
- `RSET_SUPPORT=RSET_CONSUMABLE_CPUS` seems to be needed to prevent oversubscription consequences

# Overview

- Distributed memory clusters are overall successful here.
- Some teething pains with preemption
- Paging is increasingly unsupportable.
- Disk bandwidth has suddenly become a constraint.