

UNCLASSIFIED

Accomplishments and Goals in the Age of Petascale

John T. Daly (HPC-4)
February 21, 2007

jtd@lanl.gov



Operated by the Los Alamos National Security, LLC for the DOE/NNSA

UNCLASSIFIED

LA-UR-07-1011



Acknowledgements

- LANL
 - Sarah Michalak
 - Karl-Heinz Winkler
 - Paul Iwanchuk
 - Bob Tomlinson
 - Laura Davey
 - Steve Shaw
 - John Morrison
 - Andy White
 - Joyce Guzik
 - Mark Jarzempa
 - Lori Pritchett-Sheats
 - Steve Turpin
 - Marv Alme
- Sandia
 - Sue Kelly
 - Bob Ballance
 - Jim Tomkins
 - Jon Stearley
 - John Noe
 - Mahesh Rajan
 - Courtenay Vaughan
 - Mike Davis
 - Ruth Klundt
 - Rena Haynes
 - John Aidun
 - Doug Kothe
 - Victor Kuhns
- LLNL
 - Tom Spelce
 - Richard Hedges
 - Mike Collette
 - Michel McCoy
 - Steve Louis
 - Dale Nielsen
 - Brian Carnes
 - Dave Fox
 - Kim Cupps
 - Jean Shuler
 - Barbara Herron
 - Sheila Faulkner
 - Alice Potts

Mapping Program Priorities to HPC Resources

- The NNSA Defense Program priorities list (as part of the FY2009-FY2013 Program and Resource Guidance) given with reasons for the prioritization

*4. Meet W76-1 Life Extension Program
Self-Evident*

- What resources are available for high priority work that requires a fast turnaround time (initial estimate: 16 million PE hrs in 12 weeks)?

	MANUFACTURER/COMPUTER	LOCATION		R _{max} (GFLOP/S)	PROCESSORS
TOP 5  ~1/3x PE hrs	1 IBM eServer Blue Gene Solution / BlueGene/L	Lawrence Livermore National Lab	USA	280600	131072
	2 Cray XT3 Red Storm	Sandia National Lab	USA	101400	26544
	3 IBM eServer Blue Gene Solution / BlueGene W	IBM Thomas J. Watson Research Center	USA	91290	40960
	4 IBM eServer pSeries p5 575 / ASCI Purple	Lawrence Livermore National Lab	USA	75760	12208
	5 IBM BladeCenter JS21 Cluster, PPC 970 w/Myrinet	Barcelona Supercomputer Center	Spain	62630	10240

The Initial Request for Resources

A pair of 3-D weapon calculations for the W76 Life Extension Program (LEP) system need to be **completed before March, 2007**, so the W76 LEP certification can occur on schedule. This **3-D modeling is required** to understand a specific difference between the weapon configuration used during underground nuclear testing (UGT), and the configuration being implemented during the LEP. This work will address issues included in LLNL's SRD peer review of LANL's LEP work (CODT-2006-0851). With BG/L resources, the two weapon configurations (UGT and LEP) will be modeled and will be:

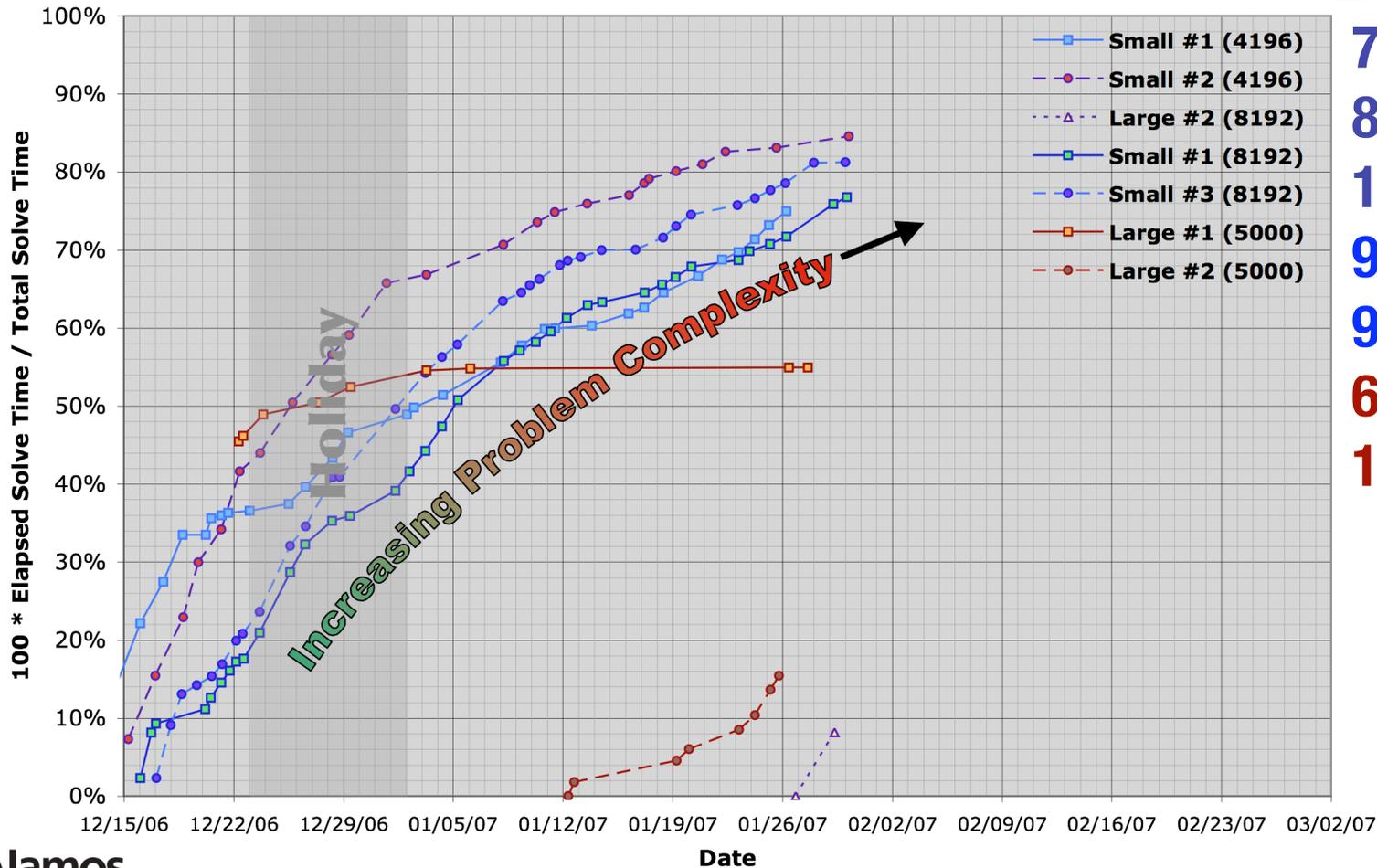
1. compared to each other;
2. compared to data from recent high-quality hydrodynamic experiments, and;
3. eventually be used as an initial state for later, more physically complicated 3-D modeling.

NOTE: The content of this message has been ADC reviewed by Joyce Guzik and determined to be unclassified.

Measuring Progress Towards the Goal

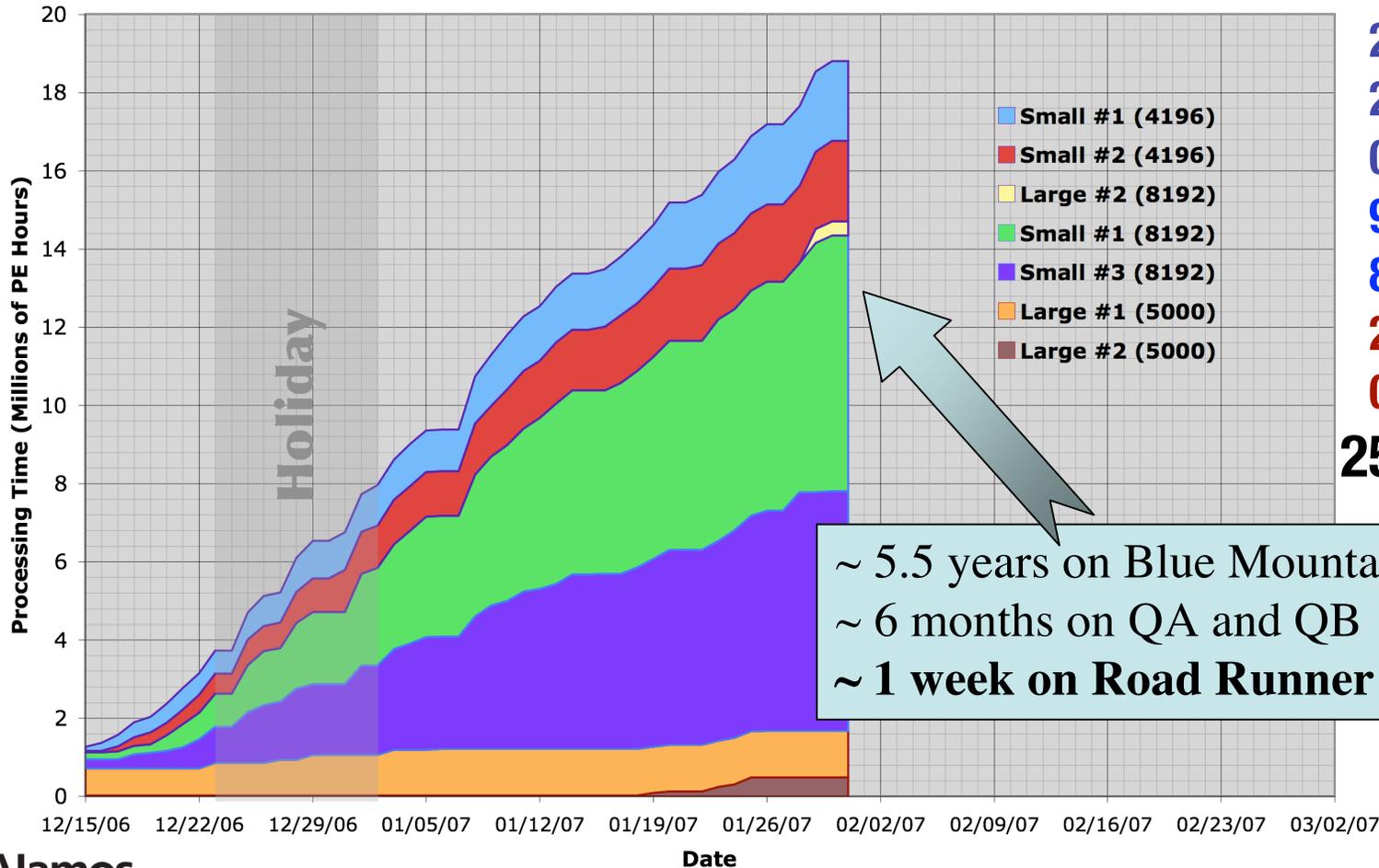
W76 LEP Progress: Percent Complete

2/20/07



How Much Compute Power is Really Needed?

W76 LEP: Cummulative Processing Time



2/20/07

2.4 Mhr

2.3 Mhr

0.8 Mhr

9.0 Mhr

8.5 Mhr

2.4 Mhr

0.5 Mhr

25.9 Mhr

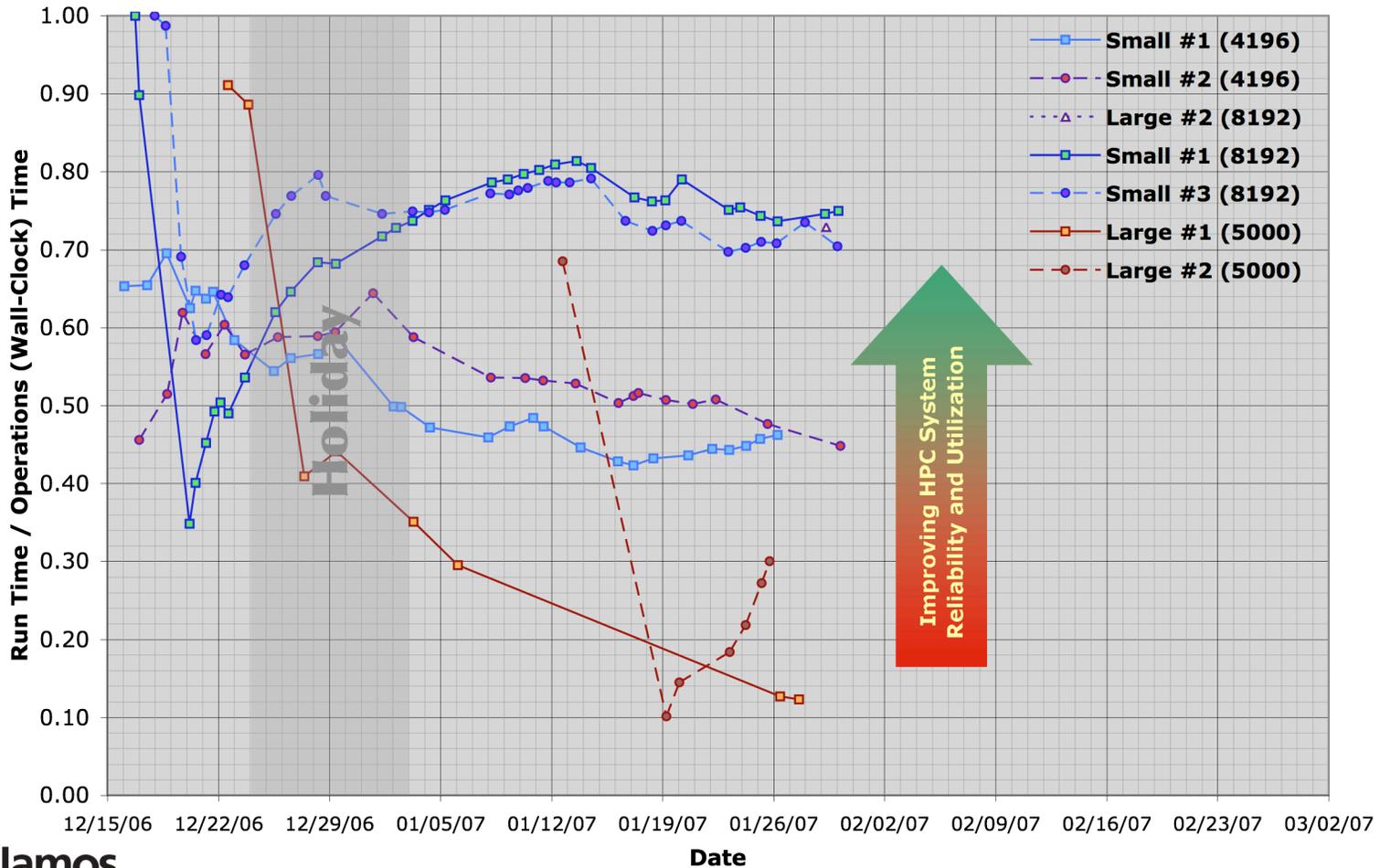
~ 5.5 years on Blue Mountain
~ 6 months on QA and QB
~ 1 week on Road Runner

How Do We Keep the Application Running?

- For the W76 LEP calculations, it is fairly typical to see about 11 interrupts per day across all platforms including
 - System hardware (nodes, disk controllers, interconnect)
 - System software (job scheduler, queue limits, file system)
 - Application errors
 - Human error
- The application is kept running by a job monitoring script
 - Kills hung jobs
 - Resubmits interrupted jobs, but *bails out for repeated failures*
 - Uses cron to restart itself even after a full system reboot
- **Goal:** If the system is available then the jobs are running!

How Efficiently are Platforms Utilized?

W76 LEP Progress: Cumulative Work Rate



Application Description

- Assumptions
 - Checkpoint rollback for fault recovery
 - Substantial restart/graphics data written periodically
 - Archival storage for future data analysis and protection against data loss
- Variables
 - Allocated Nodes: k
 - Computational Cells: C
- Application Specific Values
 - Node Performance: $P_n(k, C)$
 - Parallel Scaling: $S_p(k, C)$
 - Data Dump File Number: n_d
 - Graphic Dump File Number: n_g
- Problem Specific Values
 - Computation Work: $W(k, C)$
 - Memory Usage: $m(k, C)$
 - Data Dump Size: s_d
 - Data Dump Interval: t_d
 - Graphics Dump Size: s_g
 - Graphics Dump Interval: t_g

System Description

- Application Independent Values
 - Memory per Node: B
 - Platform Compute Nodes: N
 - Component MTBF: μ_i
 - Component MTTR: ν_i
- Application Dependent Values
 - Solver Cycle Time: $\sigma(k, P_n, S_p, W)$
 - Application MTTI: $M(k, N)$
 - Restart Overhead: $\rho(k)$
 - Idle/Reservation Time: $\kappa(k, N)$
 - Data/Restart Dump Time: $\delta(s_d, n_d)$
 - Graphics Dump Time: $\gamma(s_g, n_g)$
 - Data Archive Time: $\alpha_d(s_d, n_d)$
 - Graphics Archive Time: $\alpha_g(s_g, n_g)$
- Derived Values
 - Memory Footprint: $F(k, C | m)$
 - Productive I/O Efficiency: $E_p(\delta, \gamma, \sigma, M | t_d, t_g)$
 - Checkpoint Efficiency: $E_c(\rho, \delta, M)$
 - Operational Utilization: $U_{op}(\kappa, k | \mu_i, \nu_i, N)$
 - Production Availability: $A_{pr}(k | \mu_i, \nu_i, N)$
 - Application Throughput: $\Theta(s, E_p, U_{op})$

Requirements Definition

- Minimize Cost Function

$$\frac{\partial \Theta}{\partial \mathbf{u}} \Big|_k^N = 0 \quad \text{where} \quad \mathbf{u} = (\sigma, M, \rho, \kappa, \delta, \gamma, \alpha_d, \alpha_g | B, N, \mu_i, \nu_i, C, W, m, n_d, s_d, t_d, n_g, s_g, t_g)$$

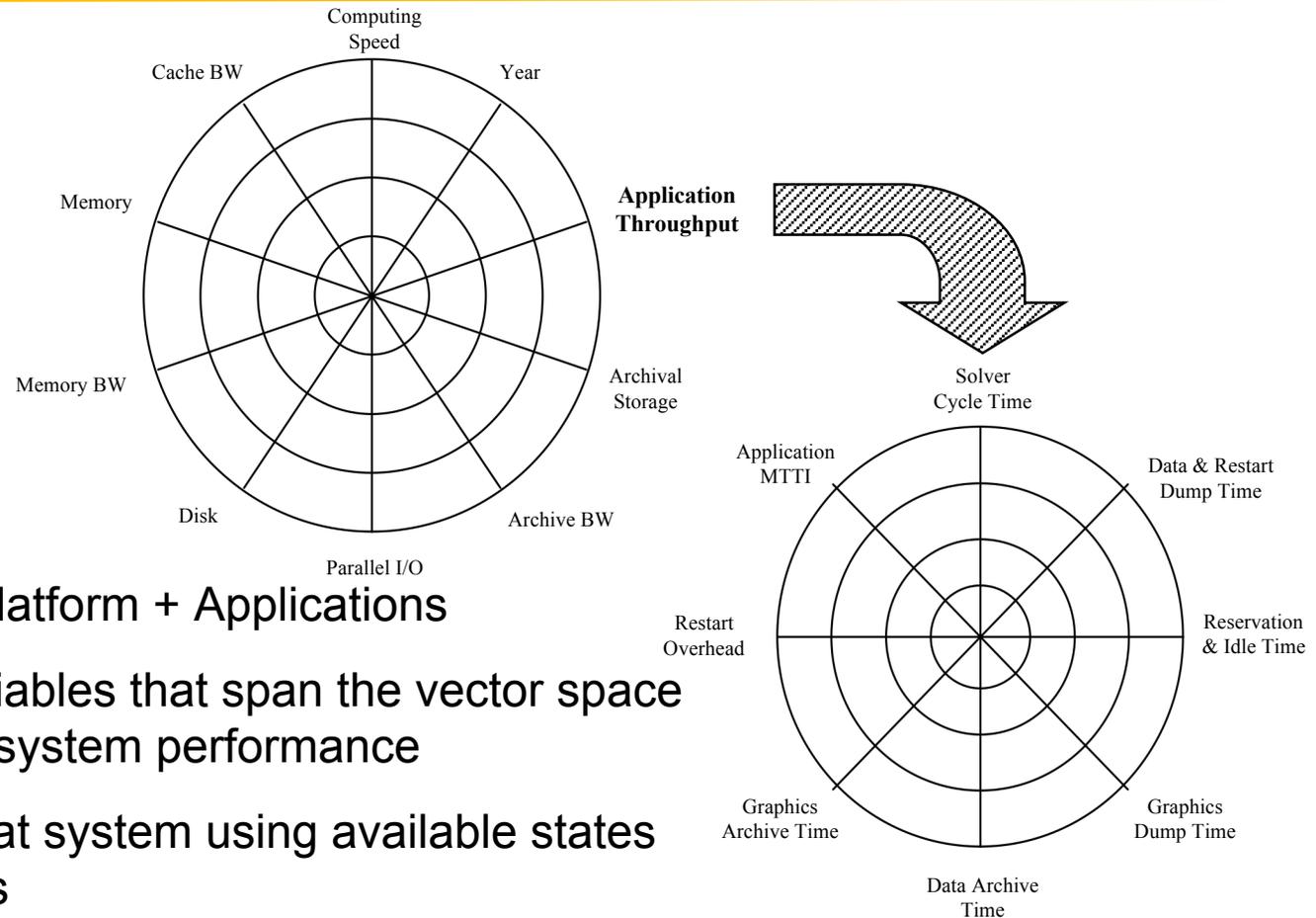
- Subject to Constraints

$$F \leq 0.75 \cdot B, \quad E_p \geq 0.98, \quad E_c \geq 0.95, \quad U_{op} \geq 0.90, \quad A_{pr} \geq 0.95,$$

$$\alpha_i \leq \begin{cases} \frac{t_i}{E_c} & \text{for limited visibility PFS (e.g. Q, Red Storm)} \\ \frac{t_i}{E_c \cdot U_{op}} & \text{for all other platforms} \end{cases} \quad \text{for } i = \{d, g\}$$

**System Requirements: Memory, Productive I/O,
Checkpoint Efficiency, Utilization, Availability,
Archival Storage**

An Application-Centric Approach to HPC



- System = Platform + Applications
- Choose variables that span the vector space of dynamic system performance
- Optimize that system using available states and controls

A Taxonomy of Application Throughput



$$\frac{1}{N} \left(\frac{W}{T_{op}} \right) = \left(\frac{W}{T_s \cdot N'} \right) \cdot \left(\frac{N}{N'} \right) \cdot \left(\frac{T_s}{T_r} \right) \cdot \left(\frac{T_r}{T_{op}} \right)$$

$$\frac{\text{Application Throughput}}{\text{Allocated Nodes}} = \text{Performance} \cdot \text{Scaling} \cdot \text{Efficiency} \cdot \text{Utilization}$$



Approximating Application MTTI

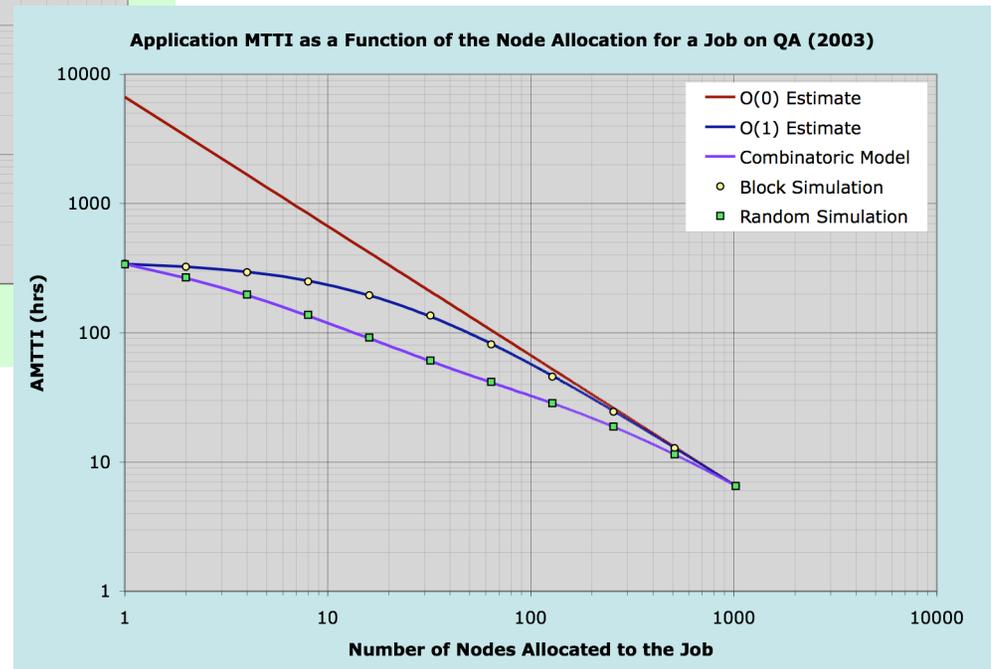
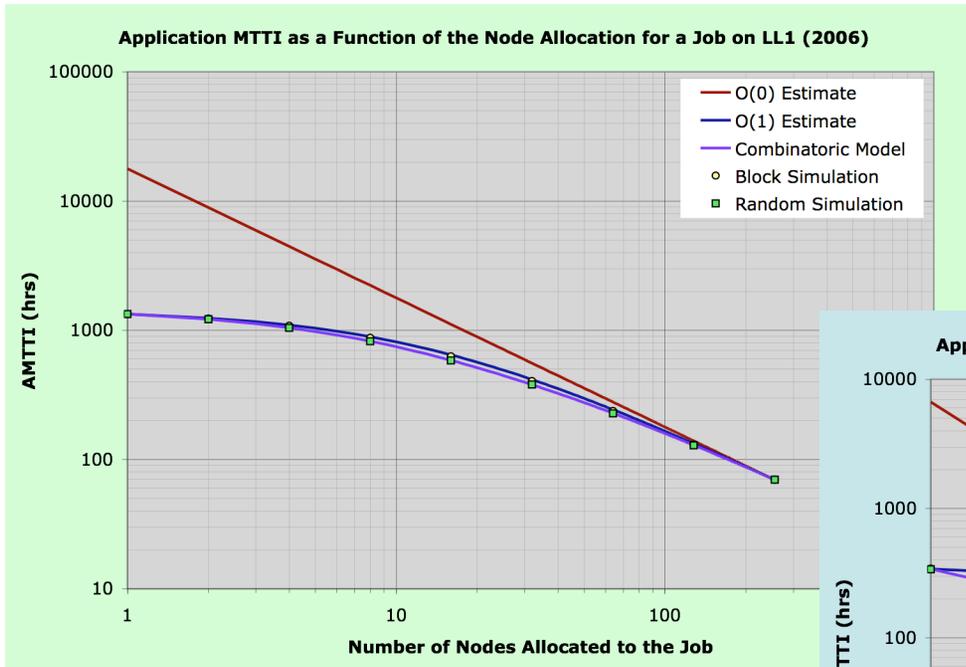
- Define μ_n as the inverse frequency associated with n node interrupts, and compute MTTI for an application running on k out of N nodes
- First, assume that for an interrupt involving k nodes, any combination of nodes is equally likely to be interrupted

$$\text{AMTTI}_k = \left(\sum_{n=1}^N \mu_n^{-1} - \sum_{n=1}^{N-k} \frac{\binom{N-k}{n}}{\binom{N}{n}} \mu_n^{-1} \right)^{-1}$$

- Next, assume interrupts occur on contiguous node blocks

$$\text{AMTTI}_k = \left(\mu_N^{-1} + \frac{k-1}{N-1} \sum_{n=1}^{N-1} \mu_n^{-1} + \frac{N-k}{N-1} \sum_{n=1}^{N-1} \frac{n}{N} \cdot \mu_n^{-1} \right)^{-1}$$

Comparing AMTTI Approximations to RAS Data

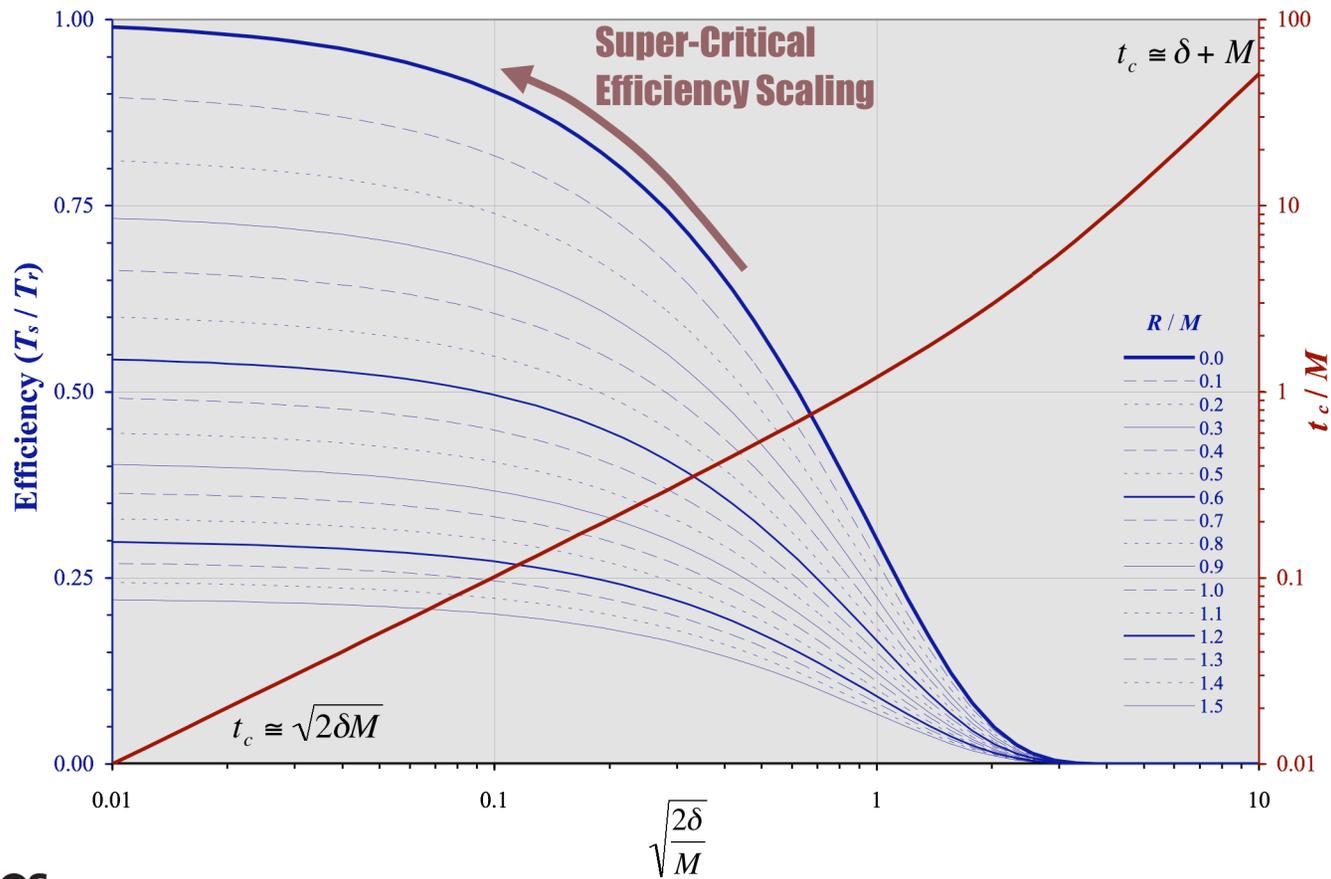


Super-Critical Efficiency Scaling (SCES)

- The scaling of application MTTI with the number of nodes is clearly sub-linear (e.g., an 8x increase in nodes may yield only a 4x decrease in AMTTI)
- If I/O rate is increasing faster than AMTTI is decreasing as more nodes are added then non-dimensional dump time ($\sqrt{2\delta/M}$) will get *smaller* (Very counter-intuitive!)
- This means that the checkpoint efficiency (E_c) is actually *increasing* with increasing numbers of nodes
- So running a few large jobs instead of many smaller ones generates higher throughput → **Capability Computing**

Scaling the Problem Up With Higher Efficiency

Checkpointing Efficiency and the Optimum Checkpoint Interval as Functions of the Dump Time, System MTBI, and Restart Overhead



Super-Linear Application Scaling (SLAS)

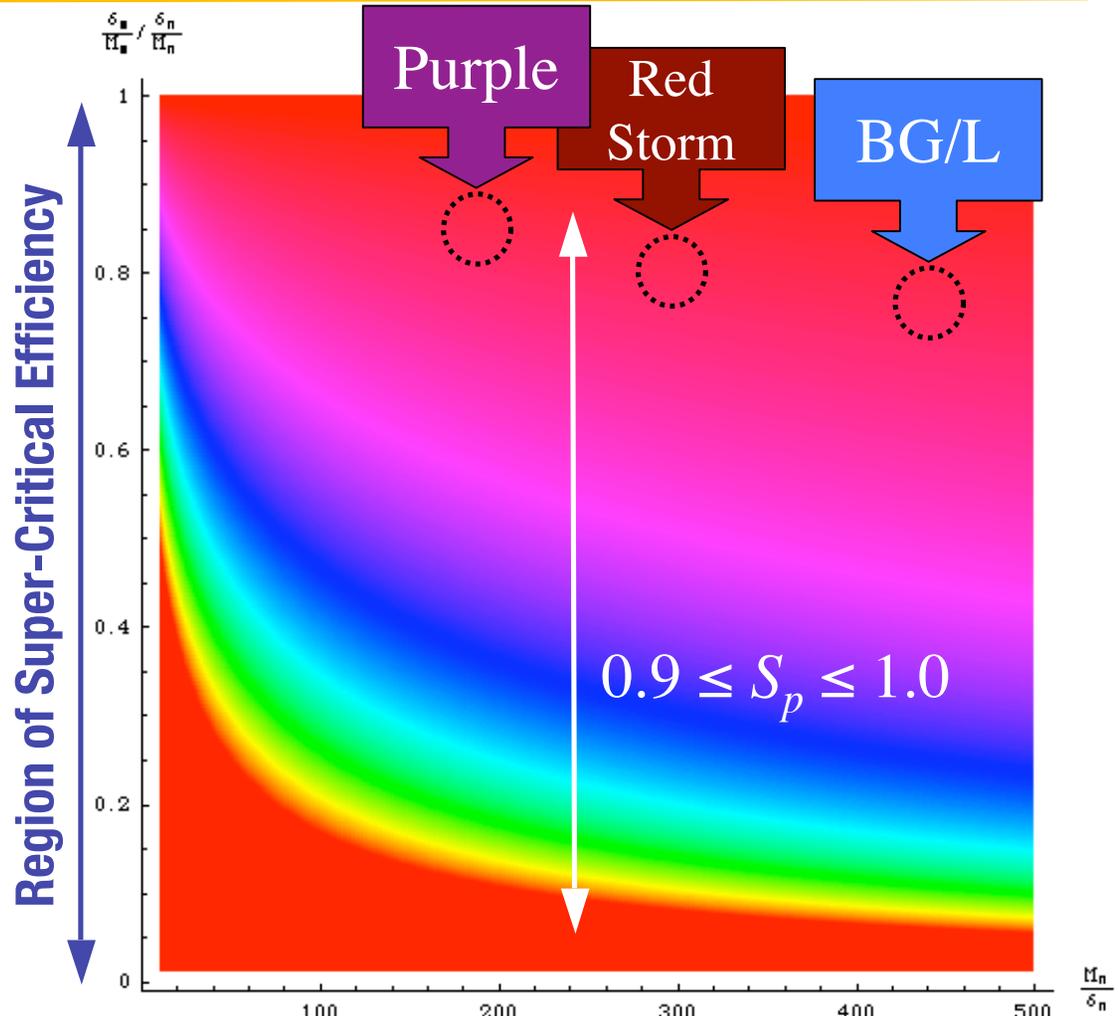
- To highest order, the checkpoint efficiency is a function of the square root of the ratio of AMTTI to dump time

$$E_c = 1 + \sqrt{2\delta/M} + O(\delta/M)$$

- If the increase in checkpoint efficiency (E_c) is greater than the decrease in parallel scaling (S_p) then running on more processors give *better* overall application throughput
- The optimum scheduling is to run jobs sequentially across the entire machine, instead of breaking them up into capability sized chunks → *application super-cruise* (ASC)
- SLAS enhances productivity for *capability* computing!

Super-Linear Scaling Machines (SLSM)

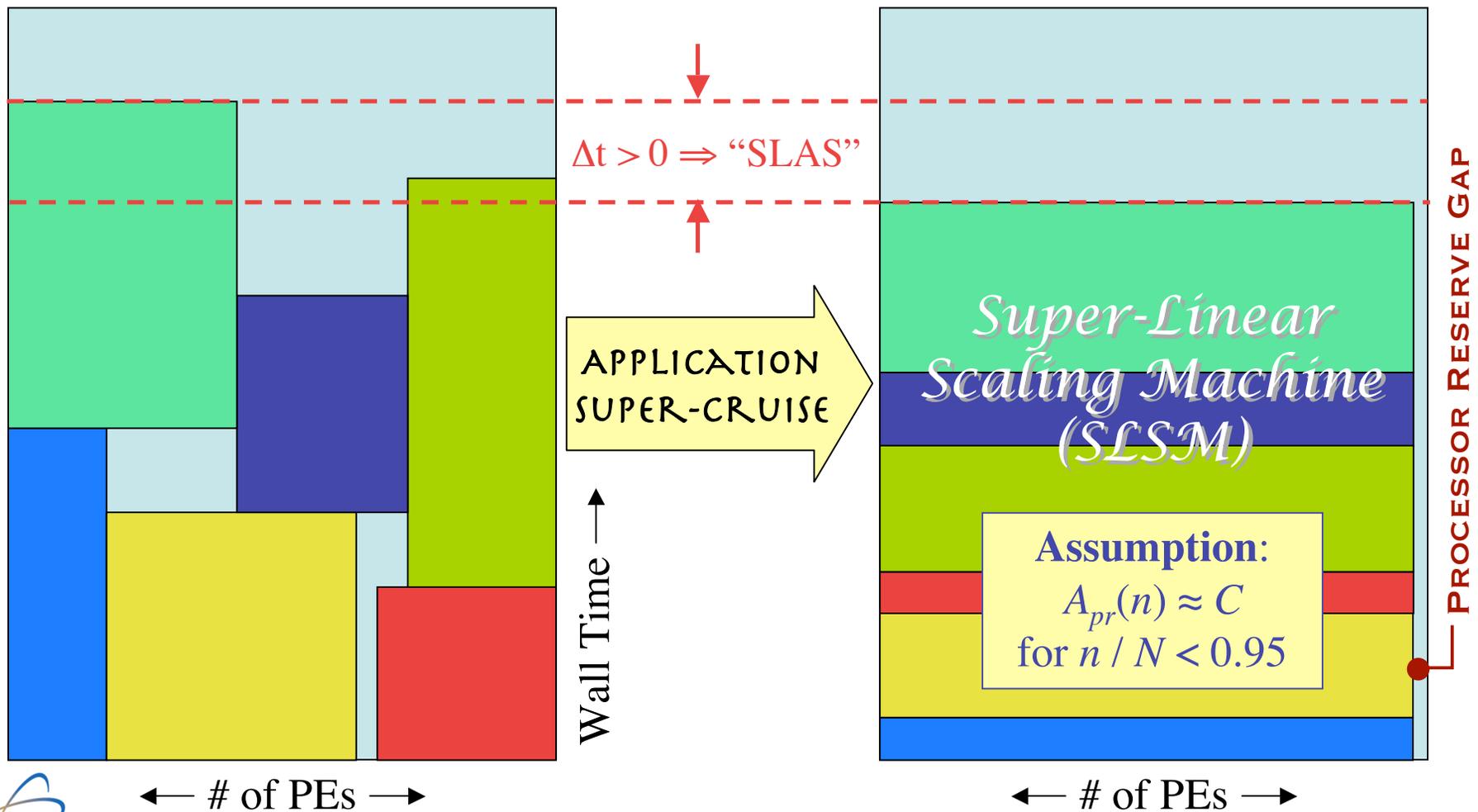
- Hues depict the minimum parallel scaling required to achieve *super-linear application scaling*
- Platform data points are notional because the data to measure them accurately was not available



UNCLASSIFIED

Slide 18

SLAS Implies a New HPC Utilization Paradigm



Conclusions

- Today's systems are adequate for many of the problems we need to solve, but not for the timescales in which we want to solve them
- Science requires a multiplicity of numerical experiments of different sizes and algorithmic complexities
- It is time to transition from *evolutionary* to *revolutionary* thinking about how we utilize HPC resources
- A radically app-centric perspective for measuring system performance will help to realize further HPC potential via novel strategies such as *super-linear application scaling*