Heuristic Static Load-Balancing Algorithm Applied to CESM

[1]Yuri Alexeev, [1]Sheri Mickelson,
[1]Sven Leyffer, [1]Robert Jacob, [2]Anthony Craig

[1]Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA
[2]CCSM Software Engineering Group, NCAR, Boulder, CO 80305, USA

Argonne National Laboratory supercomputers

Intrepid (IBM Blue Gene/P)

- 40,960 nodes / 163,840 cores
- 557 Teraflops peak
- PowerPC 450 with 4 cores/node at 850 MHz
- Double FPU – 2 wide double precision SIMD
- 512 MB per core

Mira(IBM Blue Gene/Q)

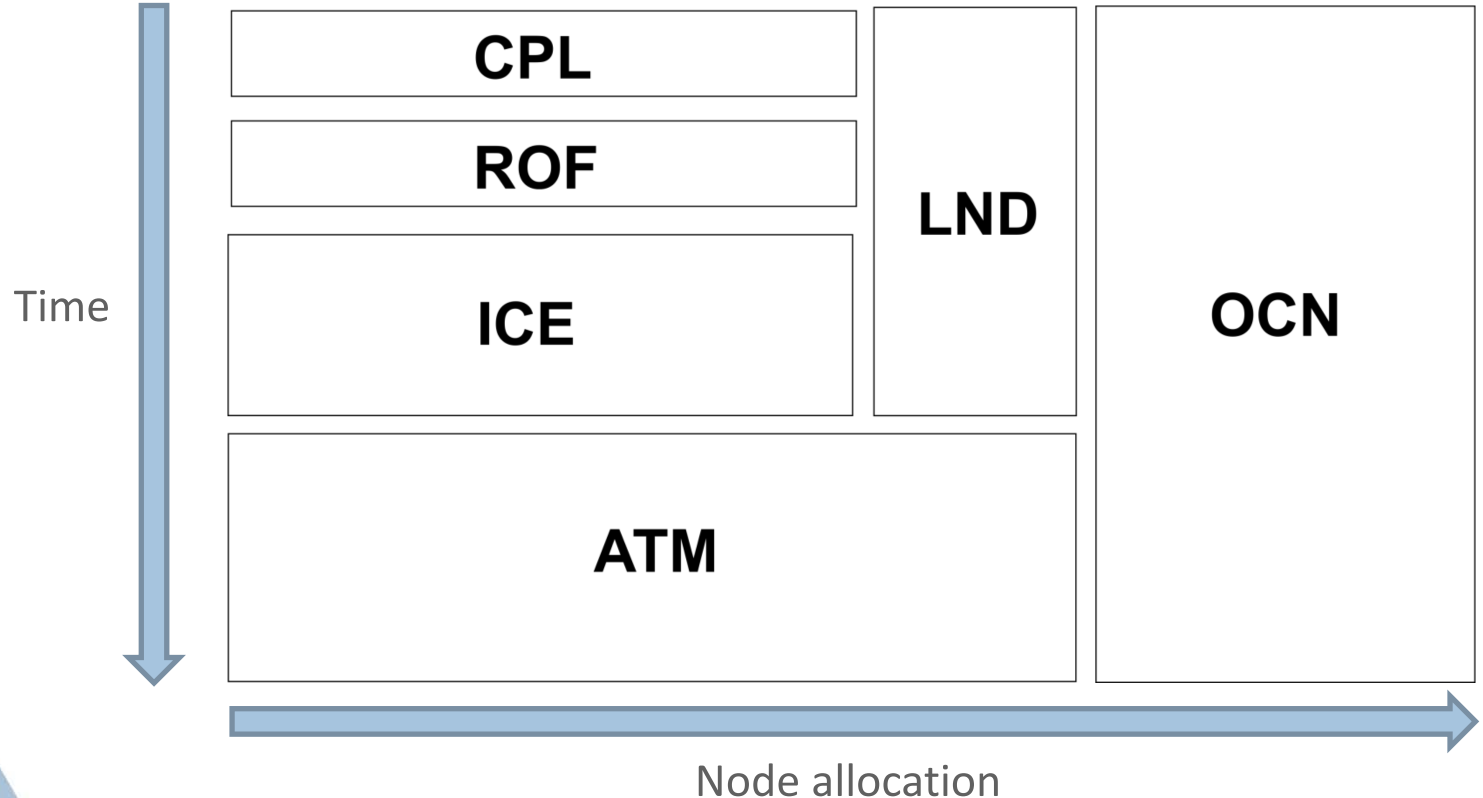
- 49,152 nodes / 786,432 cores
- 10 Petaflops peak
- PowerPC A2 with 16 cores/node at 1.6 GHz
- Quad FPU – 4 wide double precision SIMD
- 1Gb per core

CESM setup

CESM fully coupled active components, 1 degree resolution: f09_g16.B

Calculations were run on Intrepid (40 racks Blue Gene/P)

Goal: minimize total execution time



Heuristic Static Load-Balancing (HSLB) Algorithm

(1) **Gather Data:** Run CESM calculations D times using a different total numbers of cores. Collect the running times y_{ij} for each component i.

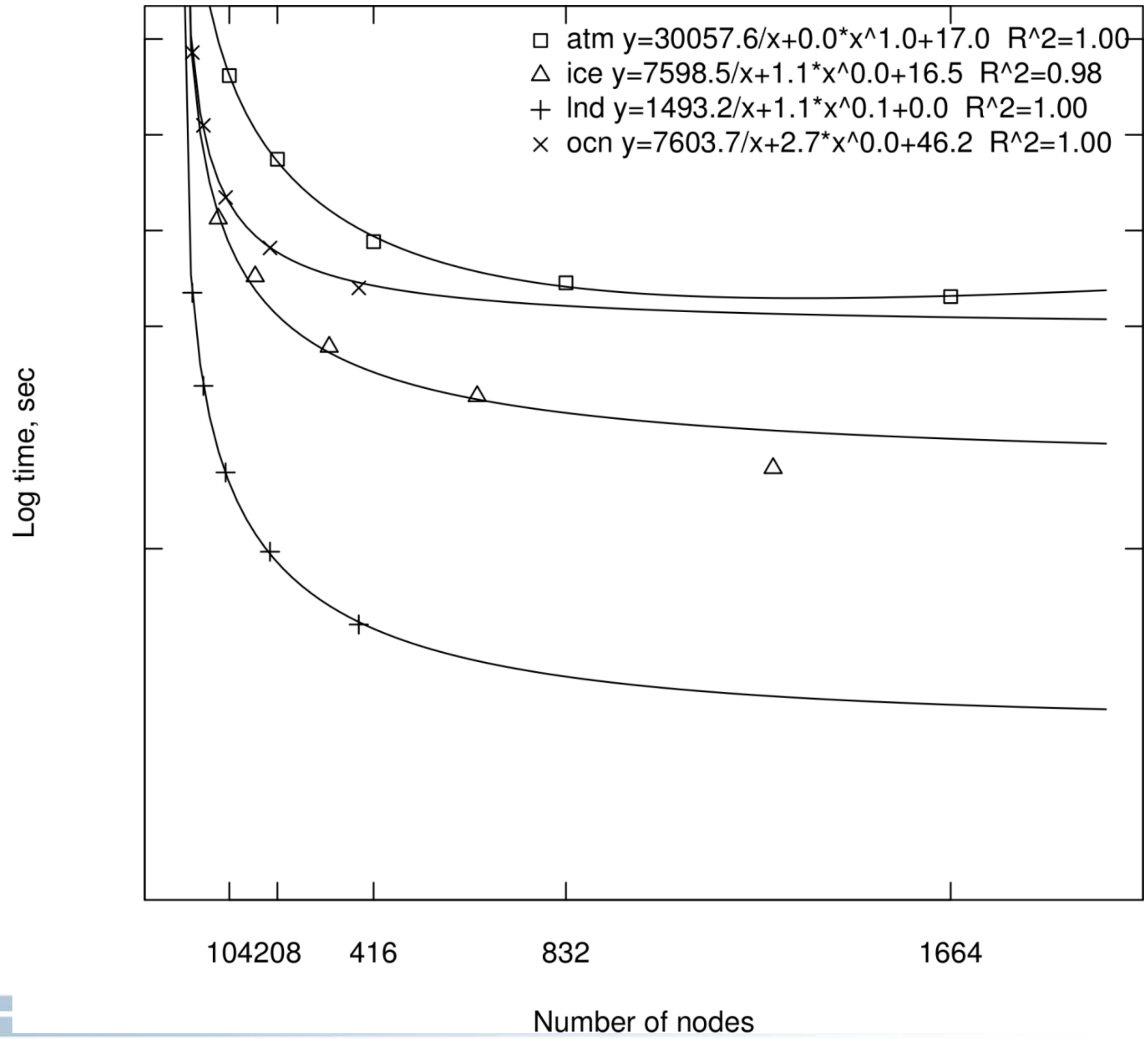
(2) **Fit:** Next, solve least squares problem for each component to determine the coefficients a_i, b_i, c_i, and d_i for each fragment i in performance model.

(3) **Solve:** Determine the best allocation by solving the MINLP, and obtain the optimal values of size n_i for each component i.

(4) **Execute:** Execute CESM simulations, using the determined subgroup sizes in step (3).

Gather data for step (1)

Calculations were run on 512, 1024, 2048, 4096, 8192 cores



Performance model for step (2)

$$T_i(n_i) = T_i^{scal}(n_i) + T_i^{nonlin}(n_i) + T_i^{serial} = \frac{a_i}{n_i} + b_i n_i^{c_i} + d_i,\ i = 1,\ldots,C$$

$T_i(n_i)$ - the wall-clock time to compute the i^{th} component as a function of n_i the number of cores allocated to process it

$T_i^{scal}(n_i) = \frac{a_i}{n_i}$ - time spent in perfectly scalable portion of the component

$T_i^{serial} = d_i$ - time spent in the non-parallelized portion of the component

$T_i^{nonlin}(n_i) = b_i n_i^{c_i}$ - time spent in partially parallelized portion: initialization, communication, and synchronization etc. (anything nonlinear and not serial)

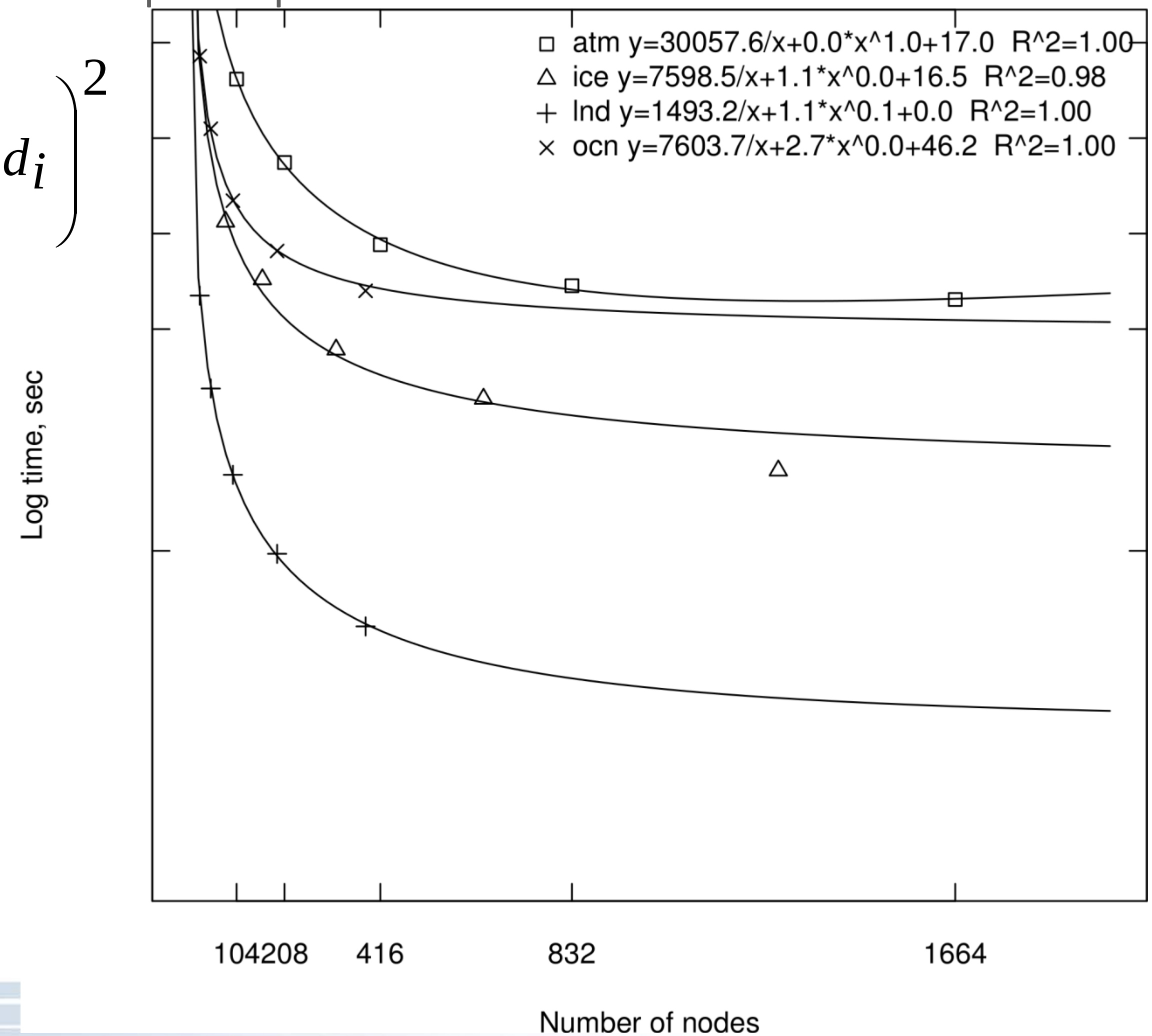
Model makes sense both mathematically and from the viewpoint of Amdahl's law

Fitting data for step (2)

Obtain the best fit by solving the least squares problem

$$\min_{a_i, b_i, c_i, d_i} \sum_{j=1}^{D_i} \left(y_{ij} - \frac{a_i}{n_{ij}} - b_i n_{ij}^{c_i} - d_i \right)^2$$

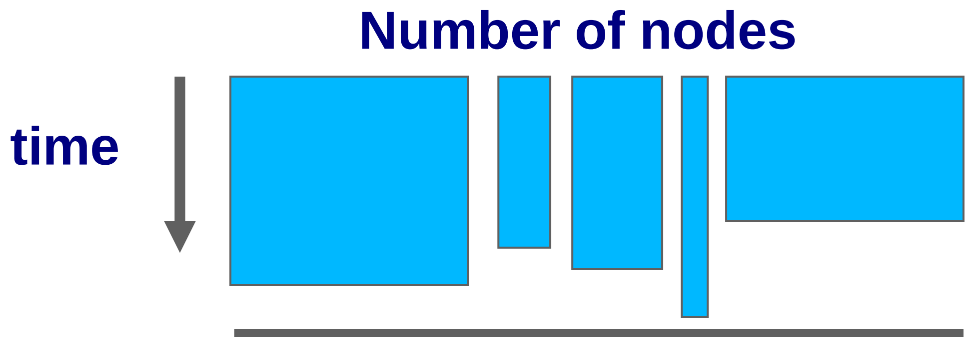
subject to $a_i, b_i, c_i, d_i \in \Re_+$



Formulating the Optimization Problem

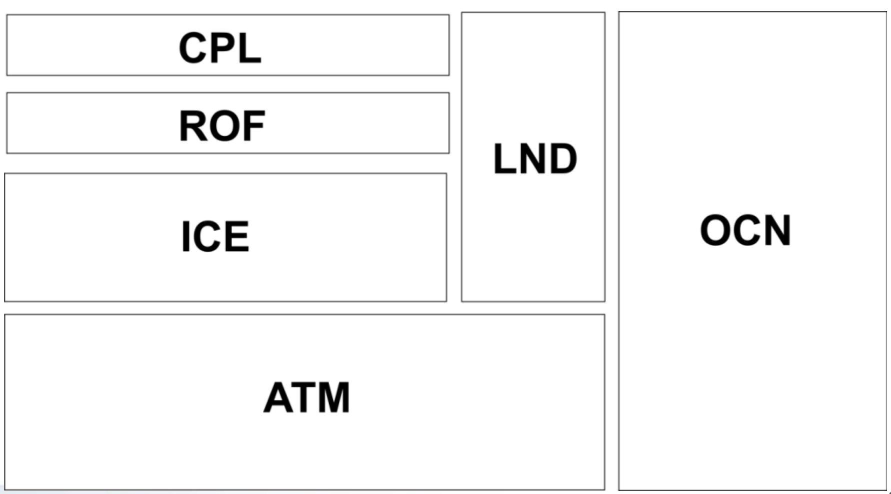
Problem: optimize the number of nodes, n_i, to be allocated to each component $i \in \{1,\ldots C\}$

- minimize the total wall time over all components : $\min\limits_{n} \sum\limits_{i=1}^{C} T_i(n_i)$
- minimize the maximum wall time used by a component : $\min\limits_{n} \max\limits_{i} T_i(n_i)$
- maximize the minimum wall time used by a component : $\max\limits_{n} \min\limits_{i} T_i(n_i)$



Formulating the mathematical problem for step (3)

1	**Given:**	$\mathbb{Z}_+$ - set of positive integer numbers
2		$\mathbb{R}_+$ - set of positive real numbers
3		$C = \{ice, lnd, atm, ocn\} = \{i, l, a, o\}$ - set of components
4		$N \in \mathbb{Z}_+$ - total number of nodes available for allocation
5		$O = \{2, 4, \ldots, 480, 768\} = \{O_1, \ldots, O_m\}$ - possible allocations for ocn
6		$A = \{1, 2, \ldots, 1638, 1664\} = \{A_1, \ldots, A_m\}$ - possible allocations for atm
7	**Variables:**	$T \in \mathbb{R}_+$ - wall-clock time obtained by solving allocation problem
8		$T_{icelnd} \in \mathbb{R}_+$ - wall-clock time to balance lnd and ice
9		$T_{sync} \in \mathbb{R}_+$ - synchronization tolerance to balance lnd and ice
10		$n_j \in \mathbb{Z}_+$ - number of nodes allocated
11		$T_j(n_j) \in \mathbb{R}_+$ - (fitted) performance function modeling time taken to run on n_j
12		$z_k \in \{0,1\}$ - binary variables to model selection of number nodes, n_o
13	**Minimize:**	T
		Constraints for layout (1)
14	**Subject to:**	$T_{icelnd} \geq T_i(n_i)$
15		$T_{icelnd} \geq T_l(n_l)$
16		$T \geq T_{icelnd} + T_a(n_a)$
17		$T \geq T_o(n_o)$
18		$T_l(n_l) \geq T_i(n_i) - T_{sync}$
19		$T_l(n_l) \leq T_i(n_i) + T_{sync}$
20		$n_a + n_o \leq N$
21		$n_i + n_l \leq n_a$



Solving MINLP problem

- Formulation is written in AMPL
- Classical branch-and-bound [Dakin, 1965] implemented in MINOTAUR:

http://wiki.mcs.anl.gov/minotaur

- Solve relaxed NLP (continuous relaxation); solution value provides lower bound

Branch on y_i

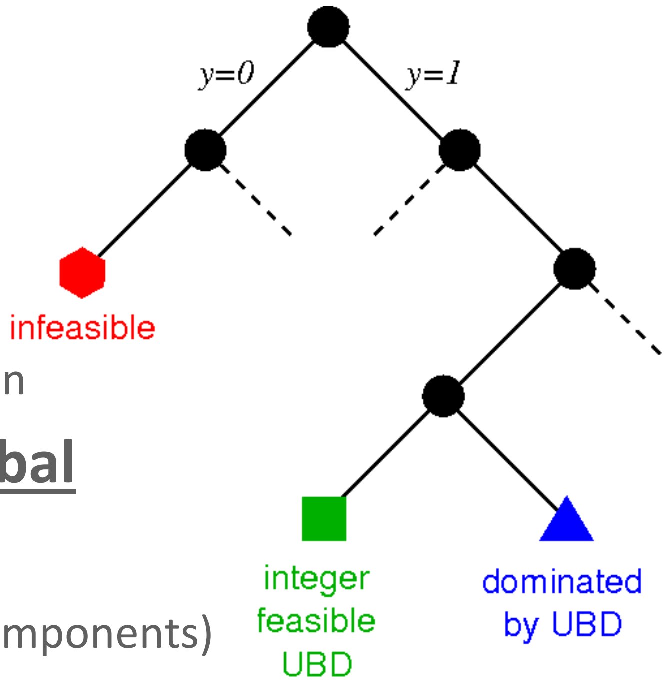
Solve NLP & branch until:

Node infeasible

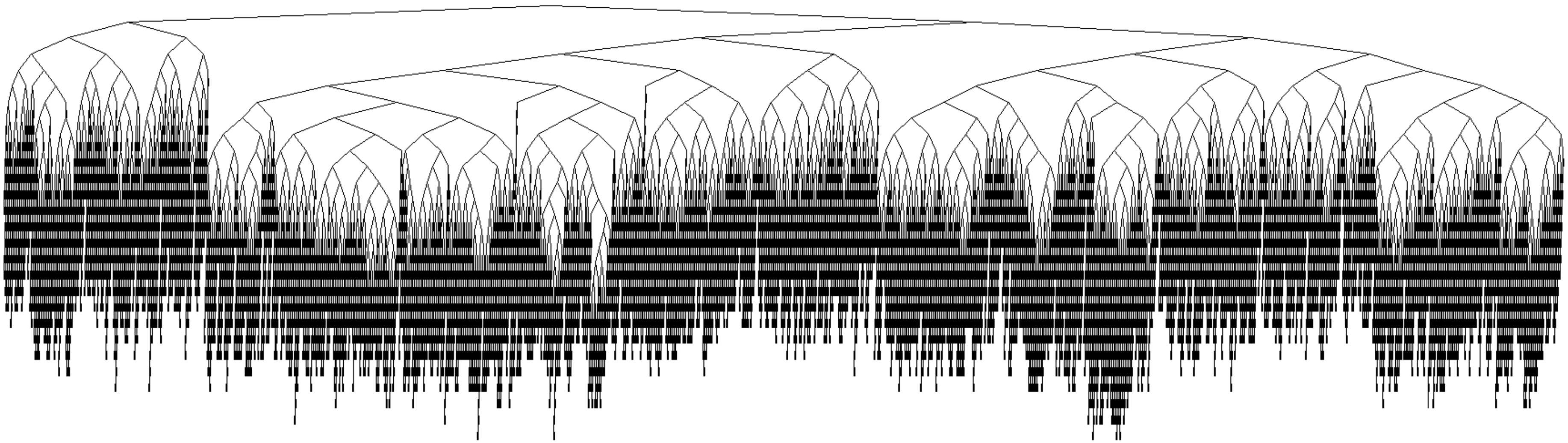
Node integer feasible (get upper bound)

Lower bound

- Tree search exhaustive but not complete enumeration
- <u>**Method guarantees to find optimal global solution or show that none exist**</u>
- Solution time is ≤ 10 seconds on a single core (155 components)



MINLP Tree



Synthesis MINLP B&B Tree: 10000+ nodes after 360s

Results

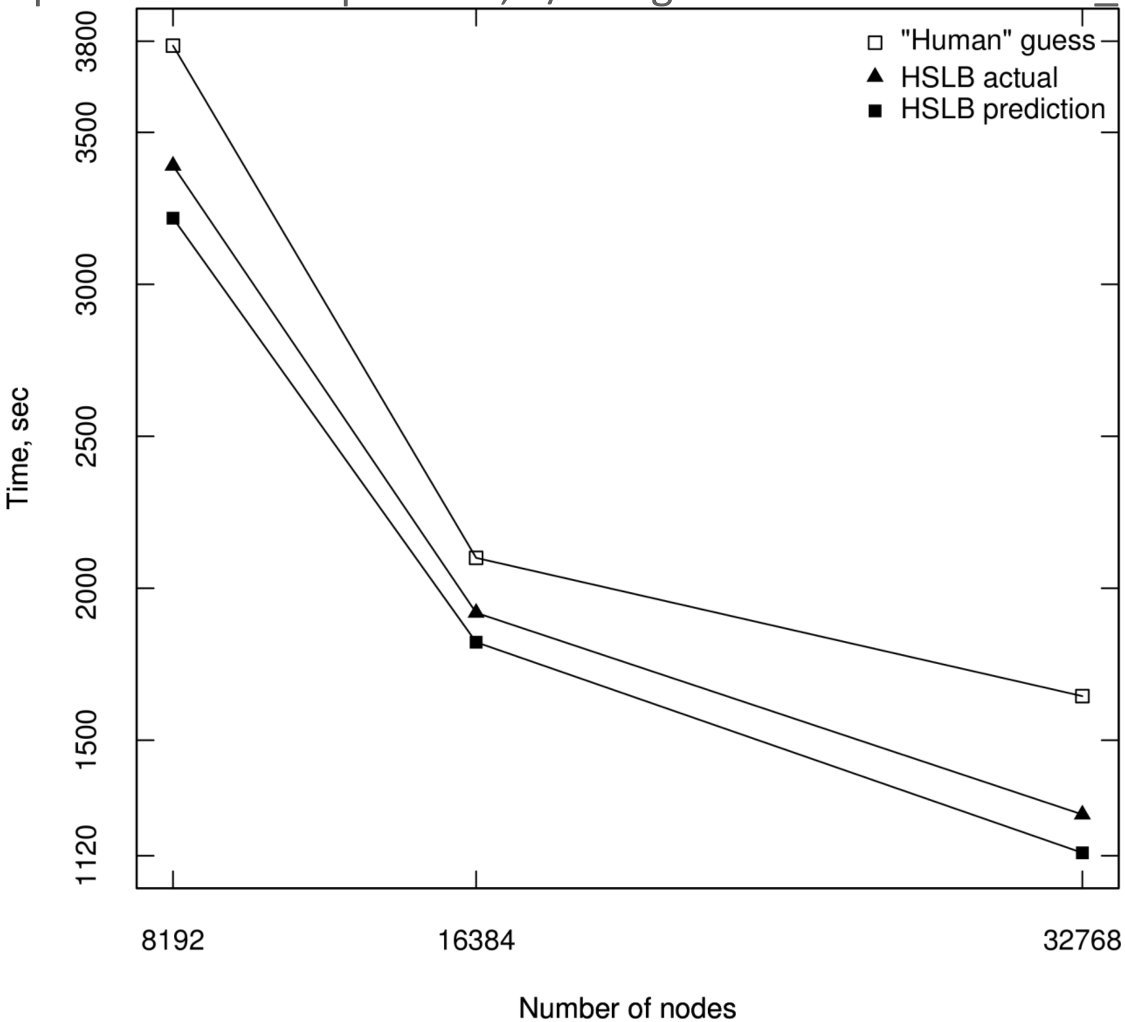
CESM fully coupled active components, 1 degree resolution: f09_g16.B

Calculations were run on Intrepid (40 racks Blue Gene/P)

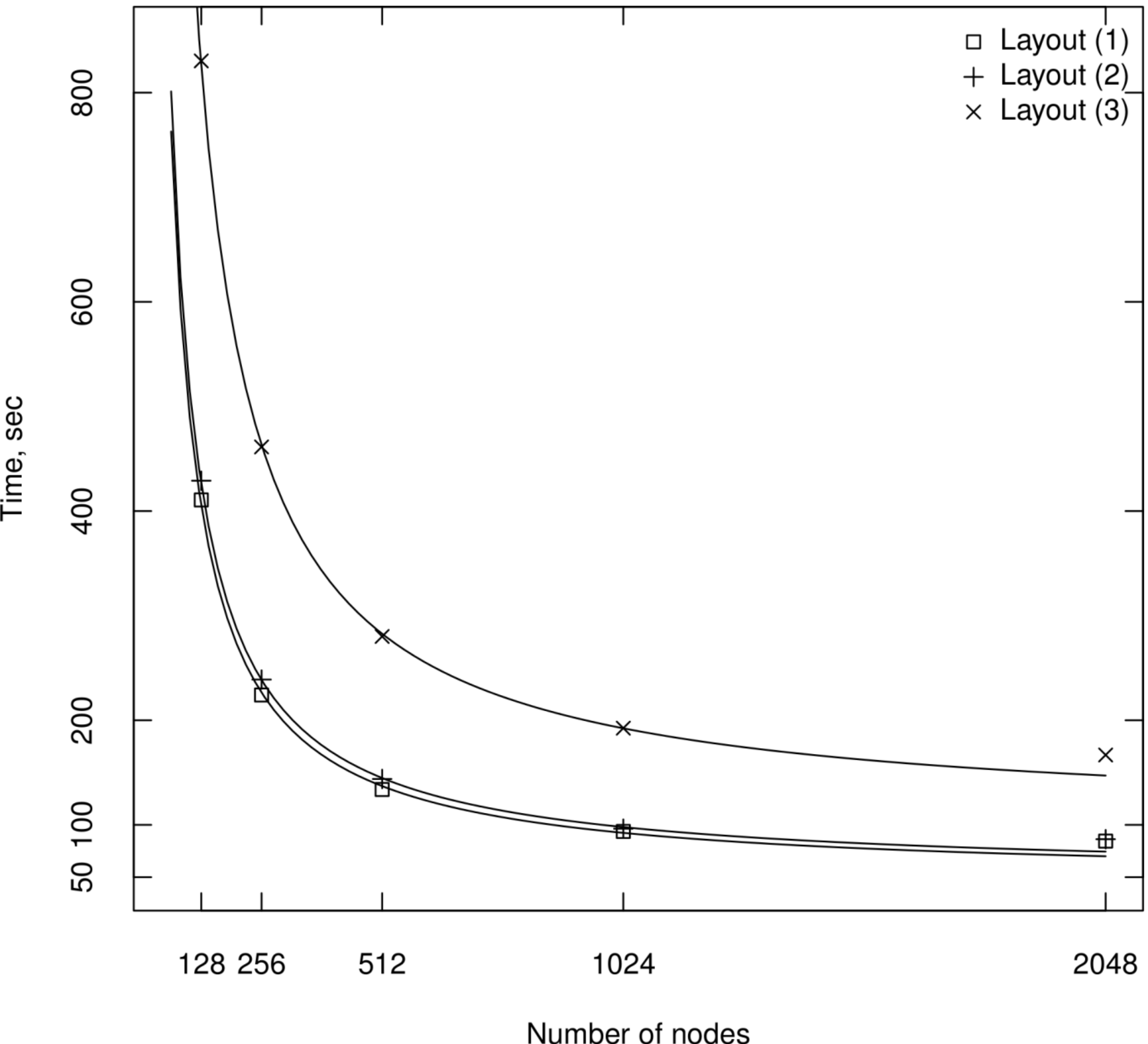
1° resolution, 128 nodes					
	Manual		HSLB		
components	# nodes	Time, sec	Predicted # nodes	Predicted Time, sec	Actual Time, sec
lnd	24	63.766	15	100.951	100.202
ice	80	109.054	89	102.972	116.472
atm	104	306.952	104	307.651	308.699
ocn	24	362.669	24	365.649	365.853
Total time, sec		416.006		410.623	425.171

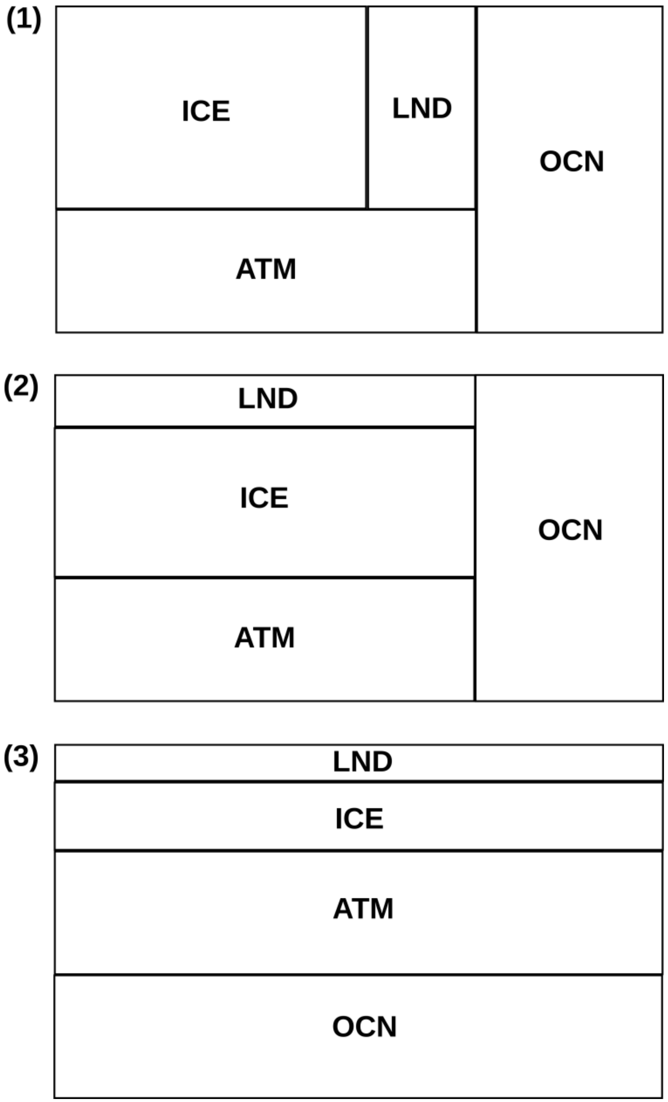
Results

CESM fully coupled active components, 1/8 degree resolution: ne240_f02_t12.B

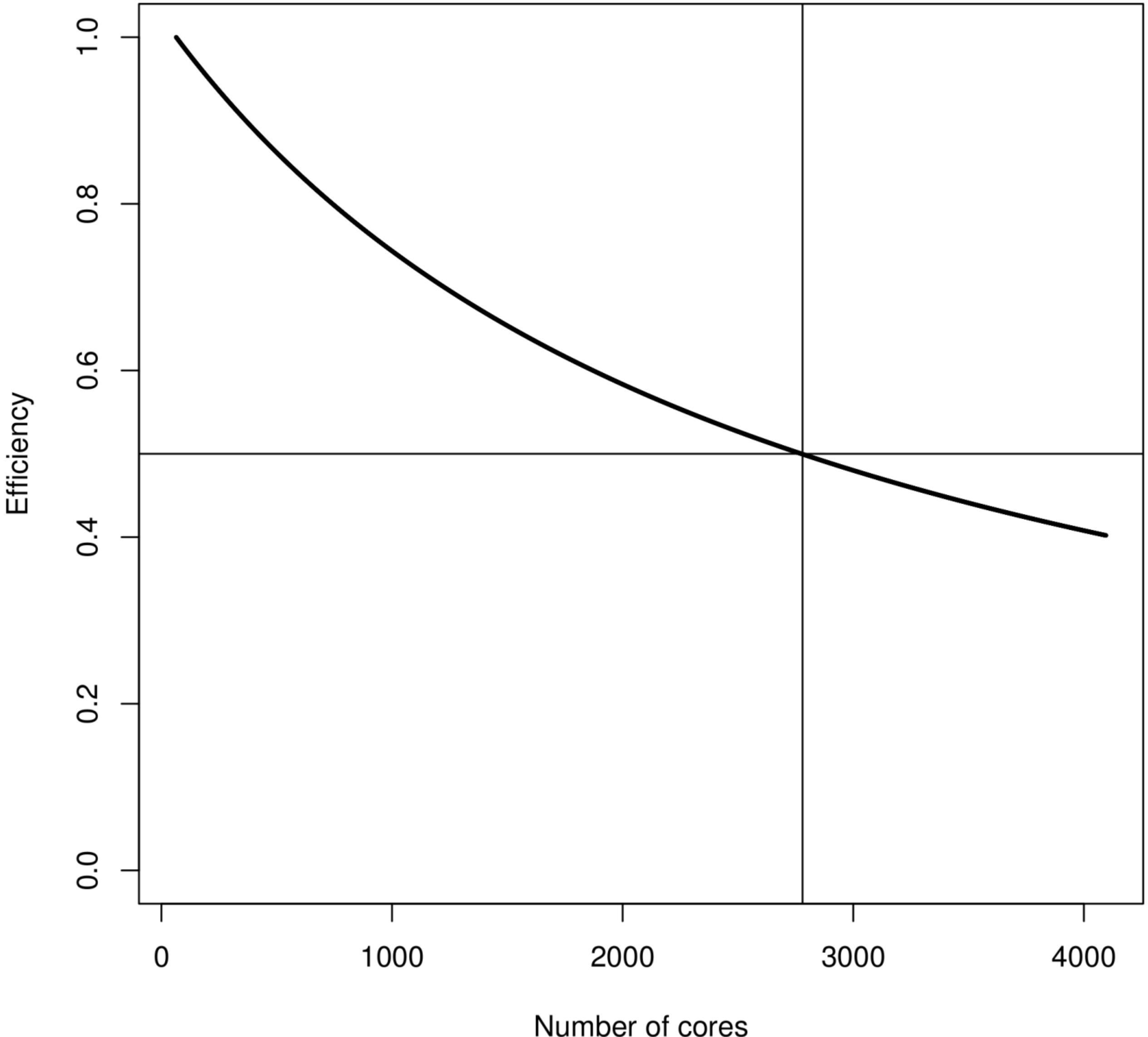


Prediction of Optimal Layout





Prediction of Efficiency



$$E(n) = \frac{T(64) / T(n)}{n / 64}$$

Future work

- Convert the AMPL code to C++ to be more portable

- Create scripts that will automate the load balancing process

- First script will gather timing data for scaling curve by creating/running 4-5 test layouts
- Second script will analyze the timing files and produce a load balanced layout based on how many cores the user would like to run on

Acknowledgments

Thank you

- Dr. Ray Loy and ALCF team members (Argonne National Laboratory)
- Jim Edwards and Mariana Vertenstein for encouraging this work and helpful discussions.

Funding was provided by

- U.S. Department of Energy