Dynamic Load Balancing for Weather Models via AMPI

Eduardo R. Rodrigues
IBM Research – Brazil
edrodri@br.ibm.com

Celso L. Mendes
University of Illinois – USA
cmendes@ncsa.illinois.edu

Laxmikant Kale
University of Illinois – USA
kale@cs.illinois.edu

Philippe Navaux
Univ.Fed.RGS - Brazil
navaux@inf.ufrgs.br

Jairo Panetta
CPTEC/INPE - Brazil
panetta@cptec.inpe.br

Outline

- Motivation
  - Performance drivers, scalability issues
  - Our approach: load balancing via Charm++/AMPI
- BRAMS Weather Model
  - Major features, capabilities
- AMPI Load Balancing for BRAMS
  - AMPI support, virtualization effects
  - Basic load balancing
- Adaptive Load Balancing
  - Balancing strategies
  - Search for optimal parameters
- Ongoing Work, Conclusion

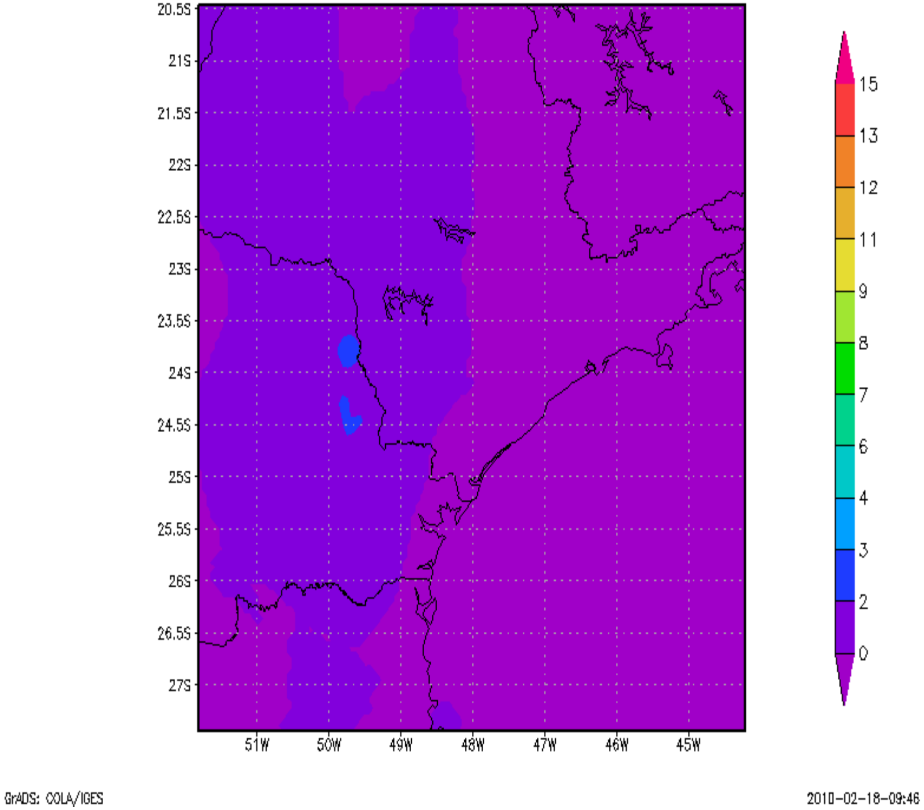
Motivation

- Climate and Weather Models
  - High computational demands
  - Typically run on large supercomputers
- Obstacles to Scalability
  - Lack of sufficient parallelism (typical 2D decomposition)
  - Overheads from communication
  - Load imbalances
    - Static sources: day/night cycle, topography
    - Dynamic sources: atmospheric phenomena
    - Typical solution: changes in model's code
      - Requires intimate knowledge of the application
      - Needs to be redone for each source of imbalance

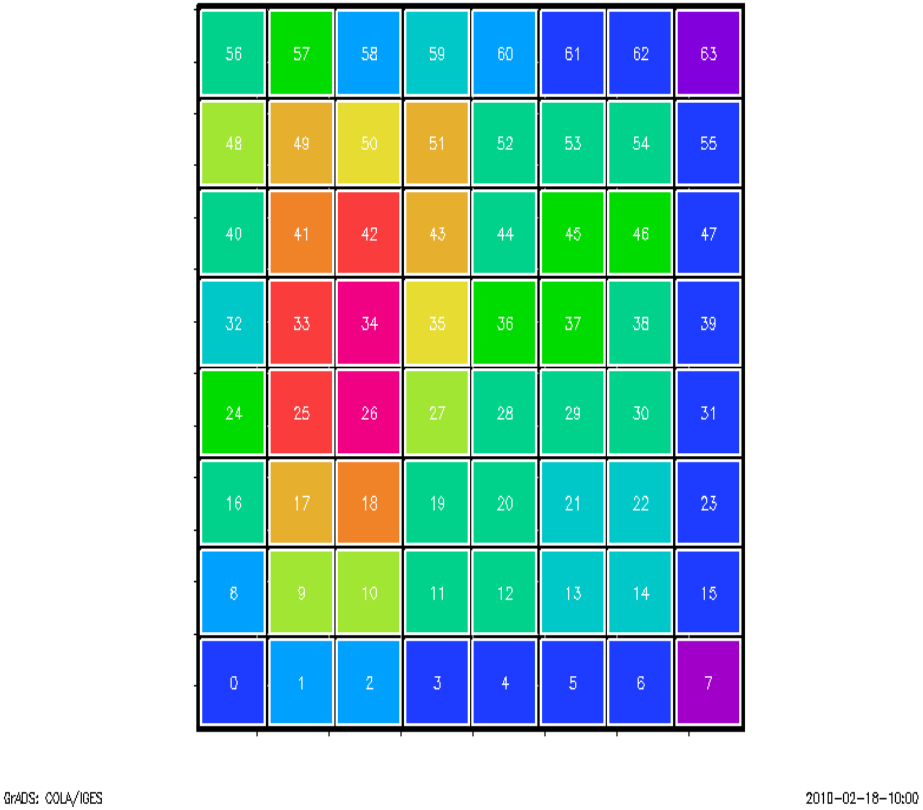
Example of Imbalance

Weather Forecast, Feb.2010

Processor Load, P=64 (8x8)



GrADS: COLA/IGES 2010-02-18-09:46



GrADS: COLA/IGES 2010-02-18-10:00

Our Approach: Charm++

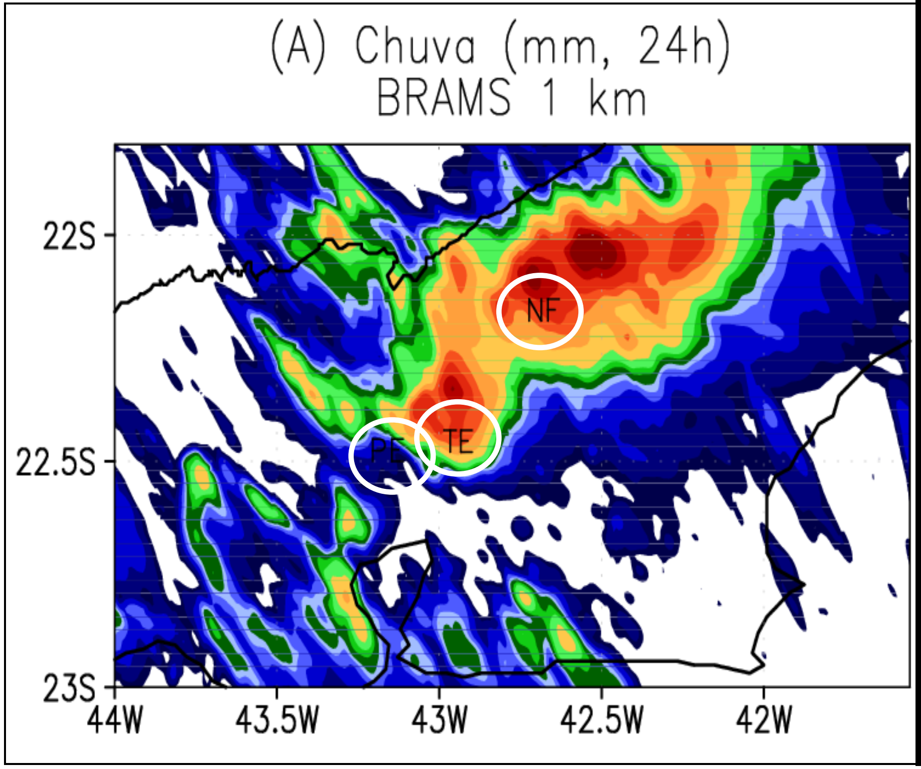
- Leverage Charm++ Run-Time System
  - http://charm.cs.illinois.edu
  - Support for MPI applications via Adaptive-MPI (AMPI)
- Charm++ Load Balancing Framework
  - Explores migration capability in Charm++
  - Balancing policies based on observed load and/or communication traffic
  - Same balancers can be applied to different codes
  - Various balancing policies available
  - Easy to create/code new policies

Case-Study: BRAMS Weather Model

- BRAMS Roots
  - RAMS, from Colorado State University
  - Model adapted for the tropics
  - Software structure modernized at CPTEC-Brazil
- Major BRAMS Features
  - Fortran90 + MPI parallelization
  - Open source, many thousands of lines of code
  - Research and production versions available
  - Support for multiple nested grids
  - Recently extended with a coupled aerosol and tracer transport model (CATT-BRAMS)
  - Daily production use, across Brazil and abroad

BRAMS Capabilities

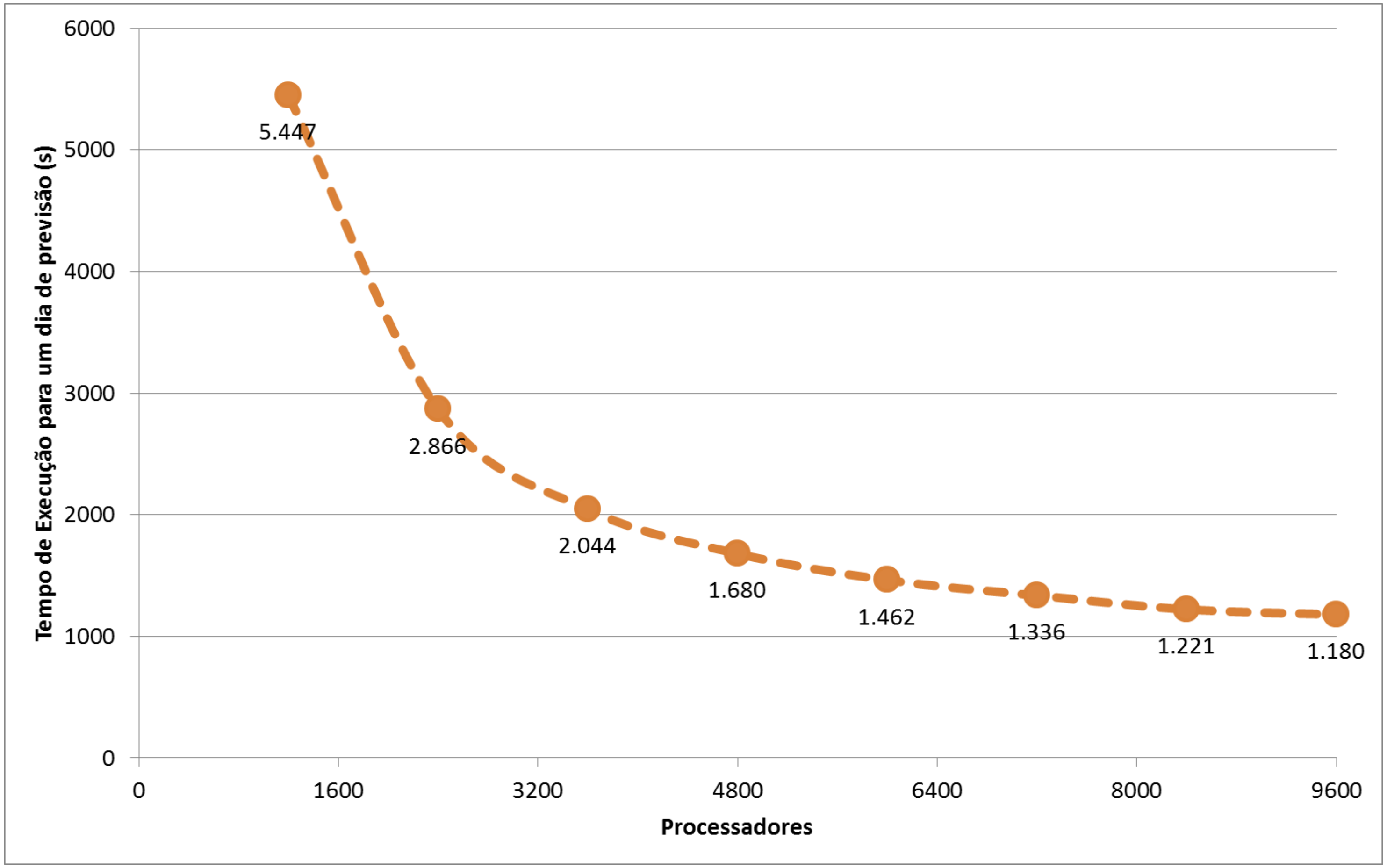
RJ-Brazil, Jan.2011 (500×500)



Accumulated Precipitation (24 hrs)		
City	Measured	Forecasted
Nova Friburgo (NF)	162	158
Teresopolis (TE)	78	88
Petropolis (PE)	7	4

Current BRAMS Scalability

Execution time for a one-day forecast, 5 Km resolution (Cray-XE6)

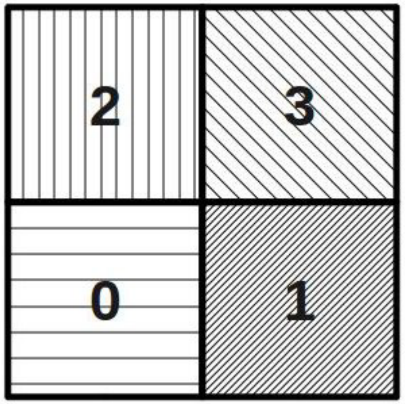


Virtualization for MPI Codes

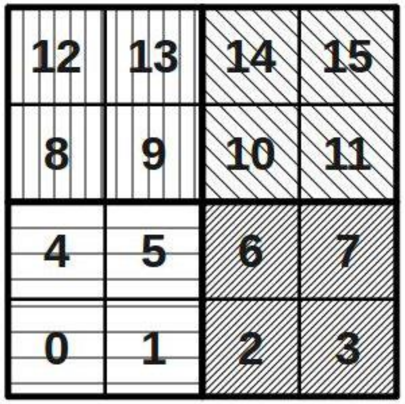
- Adaptive MPI (AMPI):
  - MPI implementation based on Charm++
  - Transforms MPI ranks into *user-level* threads
  - Processor Virtualization:
    - Multiple threads (MPI ranks) per processor
    - Mapping controlled by the Charm++ runtime

    Example: MPI code on 4 processors

MPI:



AMPI:



Porting of MPI Codes to AMPI

- Numerical Aspect
  - Code must be “numerically stable” wrt |P|
- Handling of Global/Static Variables
  - Privatize variables (similar to OpenMP regions)
  - Examples in applications:

Code	Globals	Statics
BRAMS – v.4	10,205	519
WRF – v.3	8,661	550

  - Some tools available to help in privatization
    - swapglobals flag – GOT table in ELF systems
    - TLS scheme: C/C++ __thread, patched gfortran-4.7
    - Photran’s AMPIzer – Fortran code transformation
  - Ref: Zheng et al, ICPADS’2011

BRAMS under AMPI

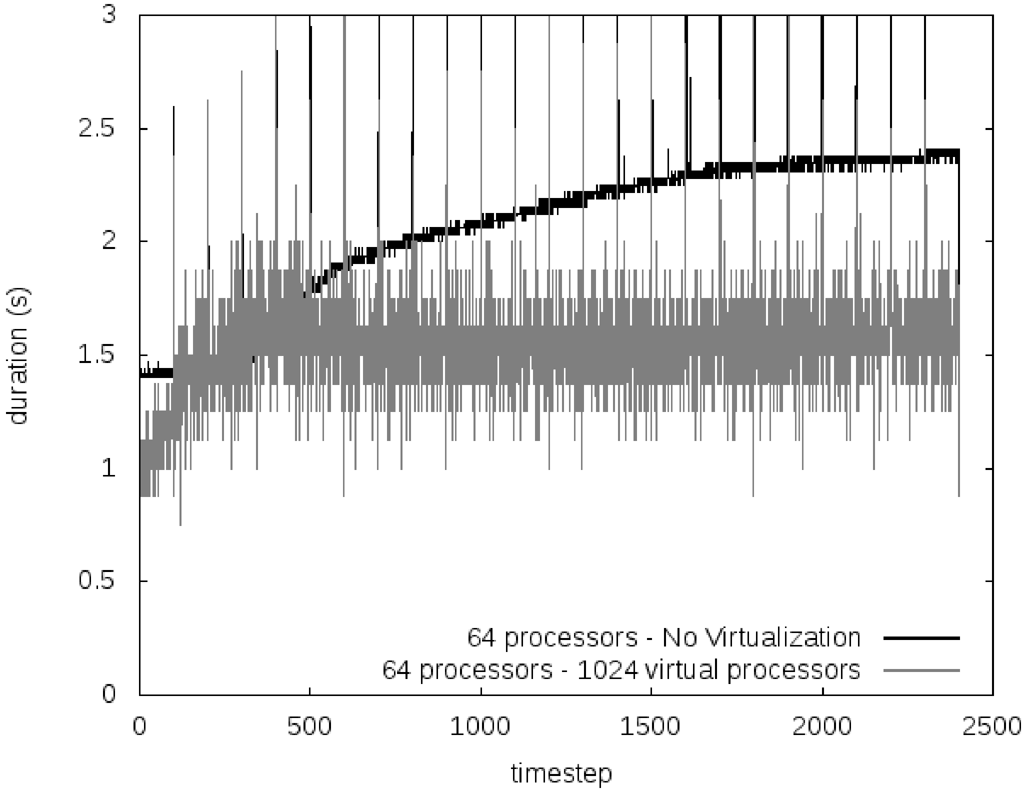
- Early Virtualization Experiments
  - BRAMS execution on 64 Kraken processors (ORNL)
    - Total grid size: 512x512x40, 1.6 Km resolution

Configuration	Execution Time (s)
64 AMPI threads (no virtualization)	4,970
256 AMPI threads	3,857
1024 AMPI threads	3,713
2048 AMPI threads	4,437

  - Two sources of performance gains
    - Overlap of computation and communication
    - Improved cache utilization
      - Better cache behavior measured via PAPI
      - Sweet-spot at 1024 threads: virtualization ratio = 16

BRAMS under AMPI

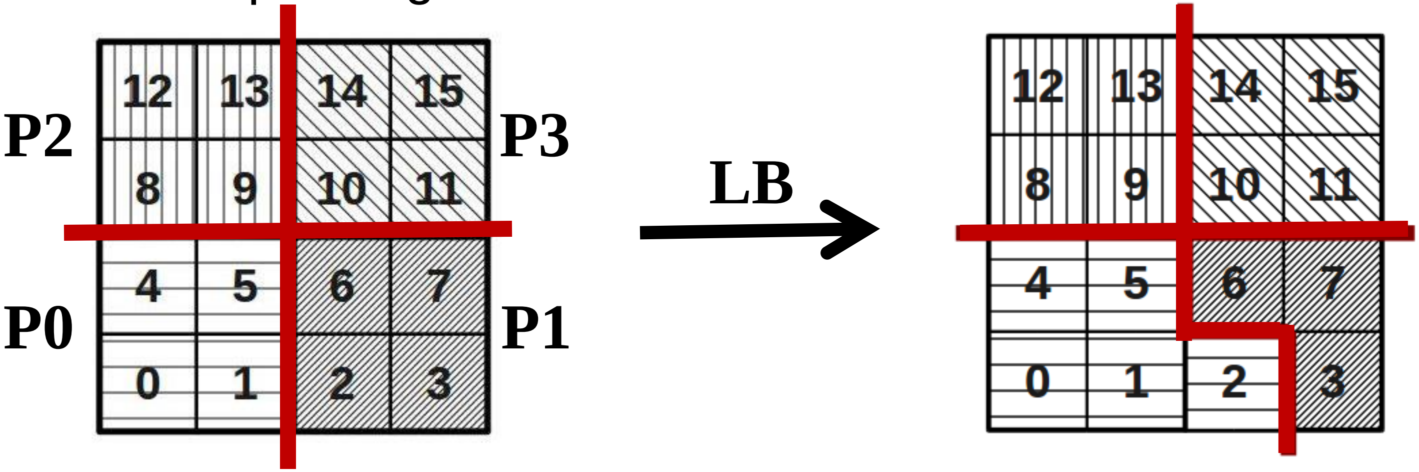
- Virtualization Effect on Timesteps
  - BRAMS execution on Kraken: P=64, VP=1024



  - Ref: Rodrigues et al, HiPC'2010

Load Balance via AMPI

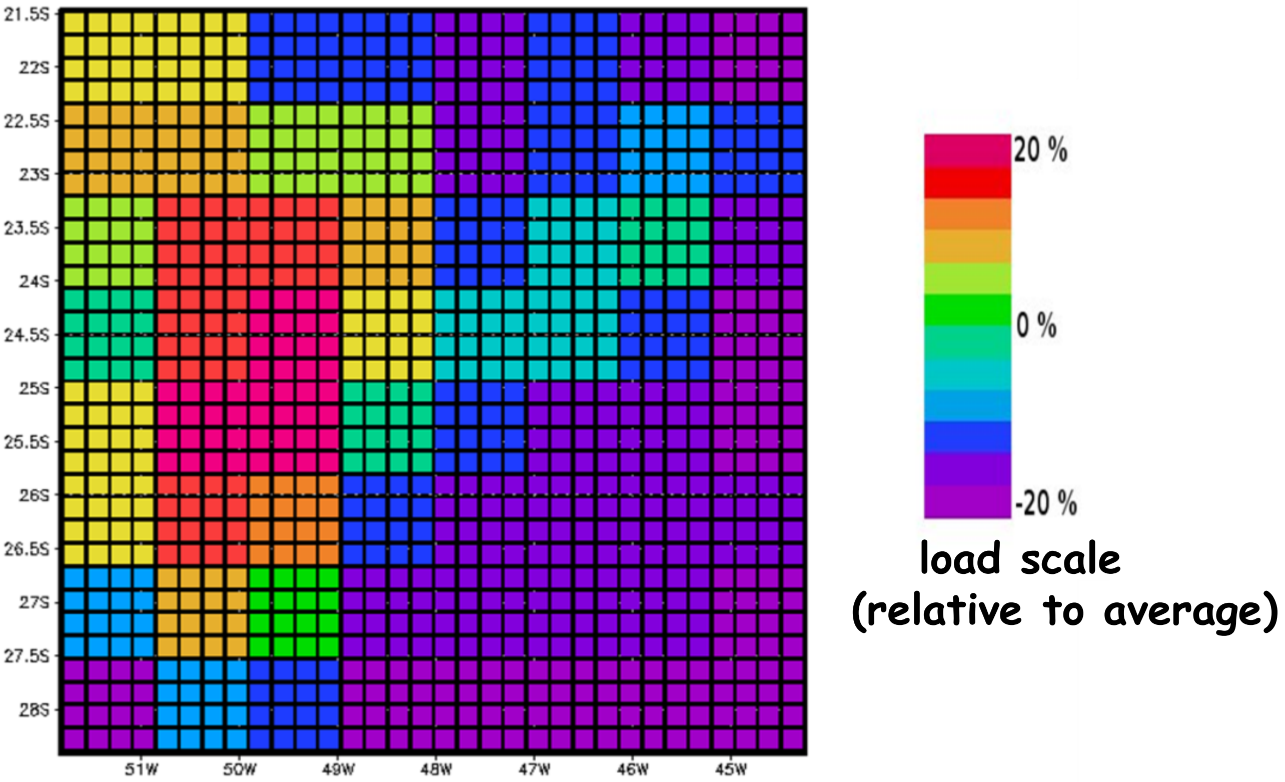
- Key AMPI Feature: Migration
  - May migrate threads across processors during execution to enable load balancing (LB)
    - Example: migration of thread 2 from P1 to P0



  - Dynamic, measurement-based load balancing
    - Balancing/migration points: call MPI_Migrate()
    - Balancing policy can be chosen at execution time
    - Measurements can be turned on/off in given phases

AMPI's Load Balance in BRAMS

- Load distribution after 600 tsteps
  - P=64, 1024 AMPI threads (≡ 1024 MPI ranks)



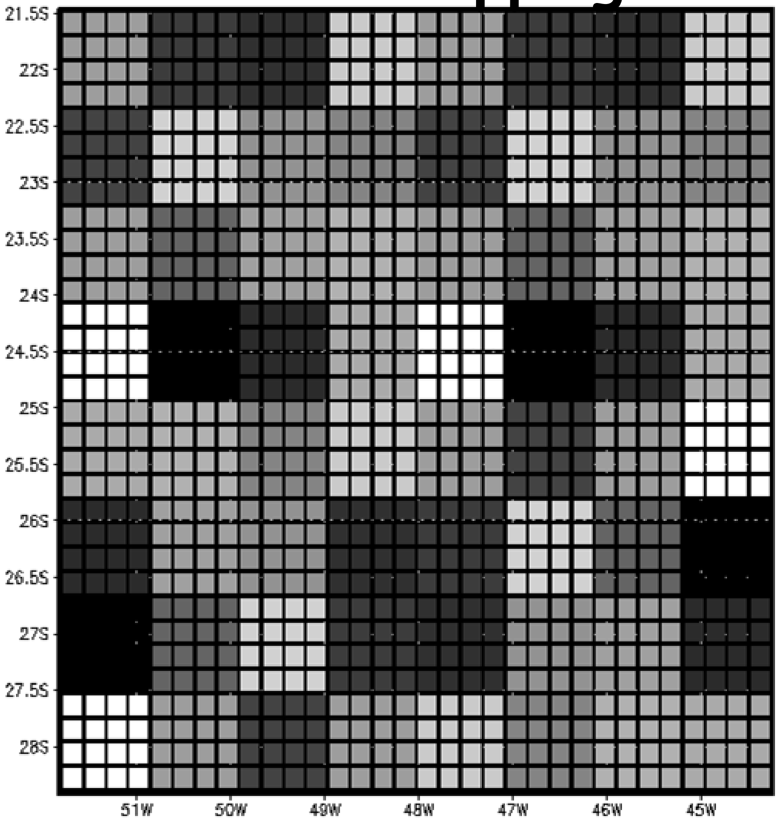
load scale
(relative to average)

AMPI’s Load Balance in BRAMS

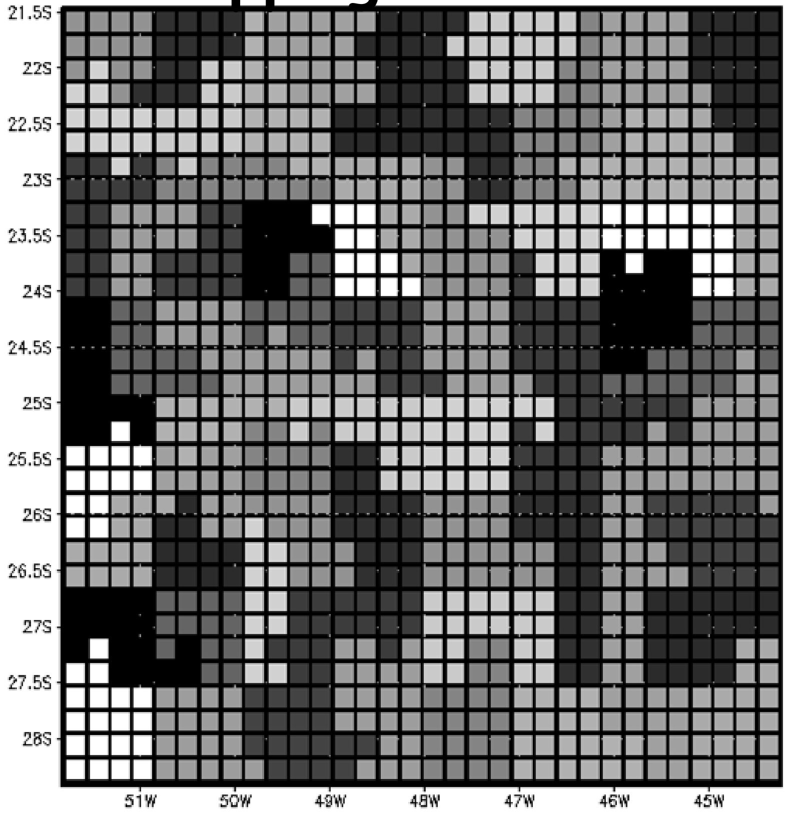
- Mapping of Threads Before/After LB

(Random shades used here, for illustration only)

initial mapping

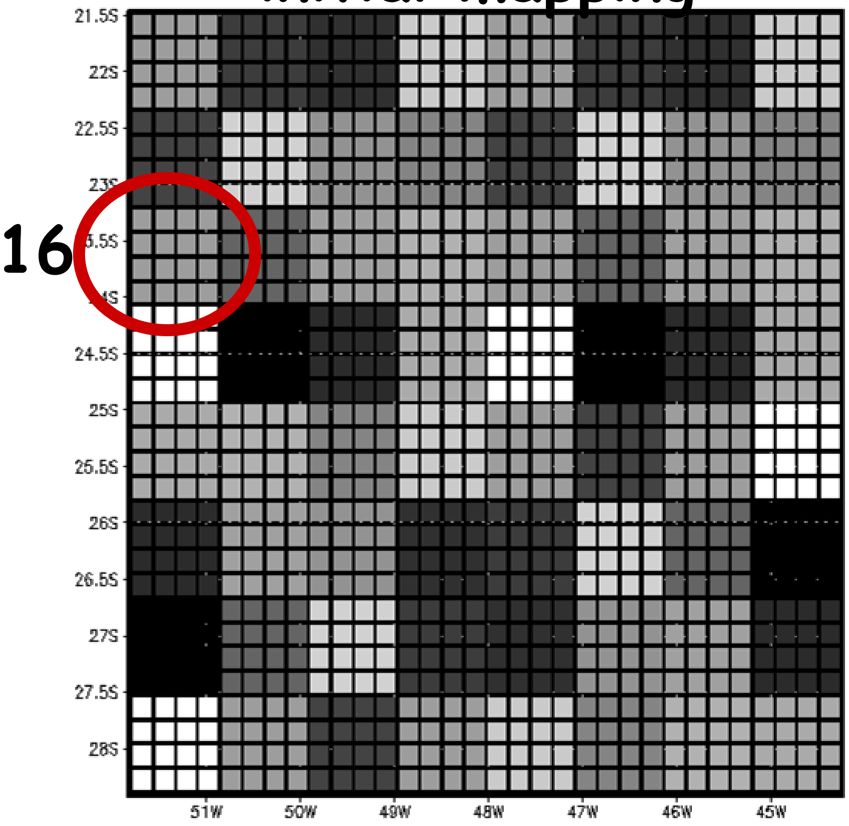


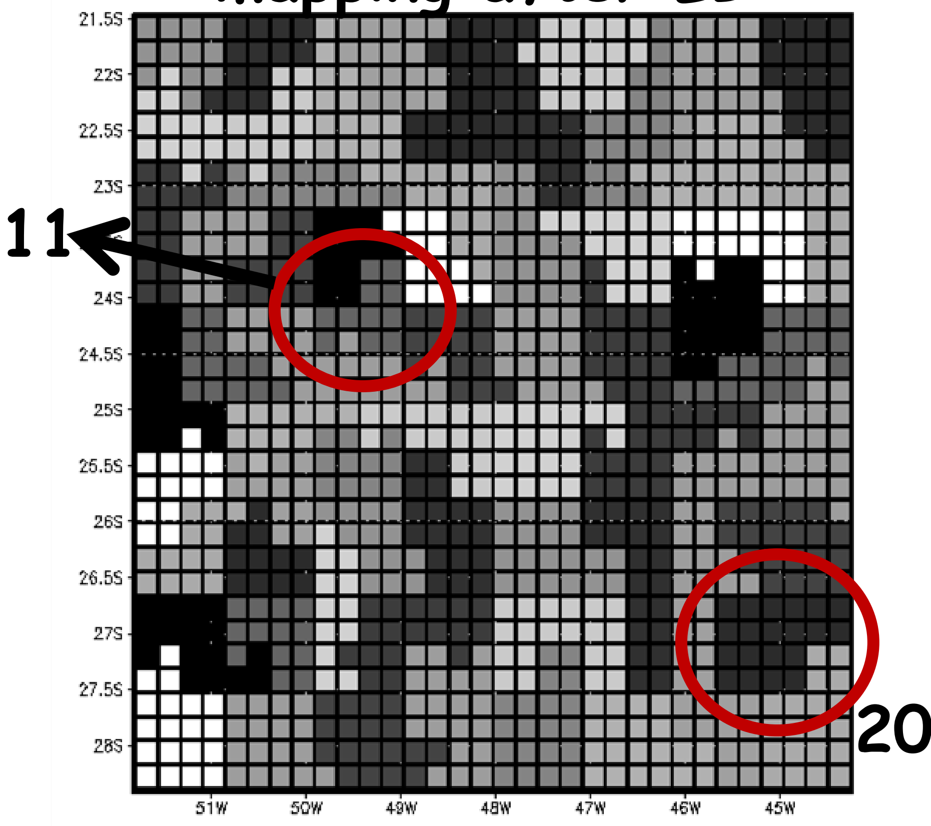
mapping after LB



AMPI's Load Balance in BRAMS

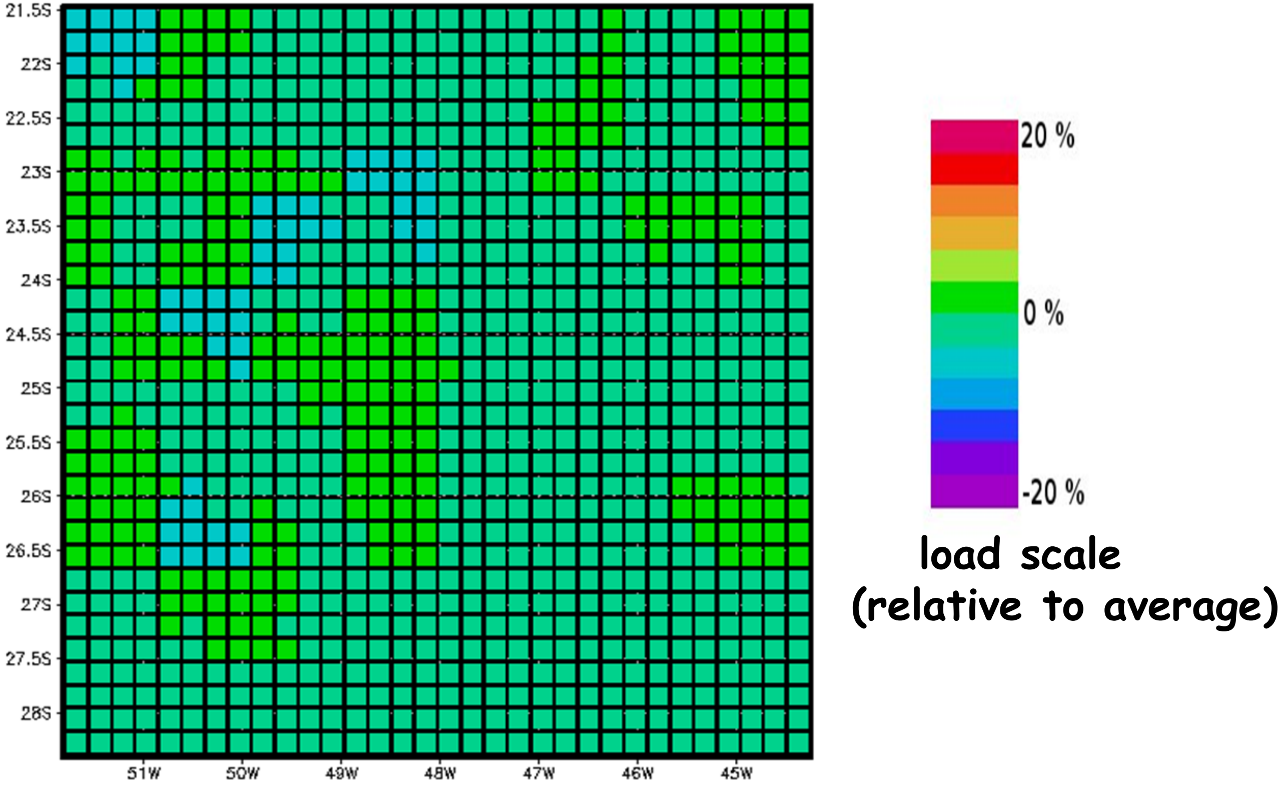
- LB Leads to Variable #Threads/Proc
  - Such that **load** is uniform across processors


initial mapping


mapping after LB

AMPI's Load Balance in BRAMS

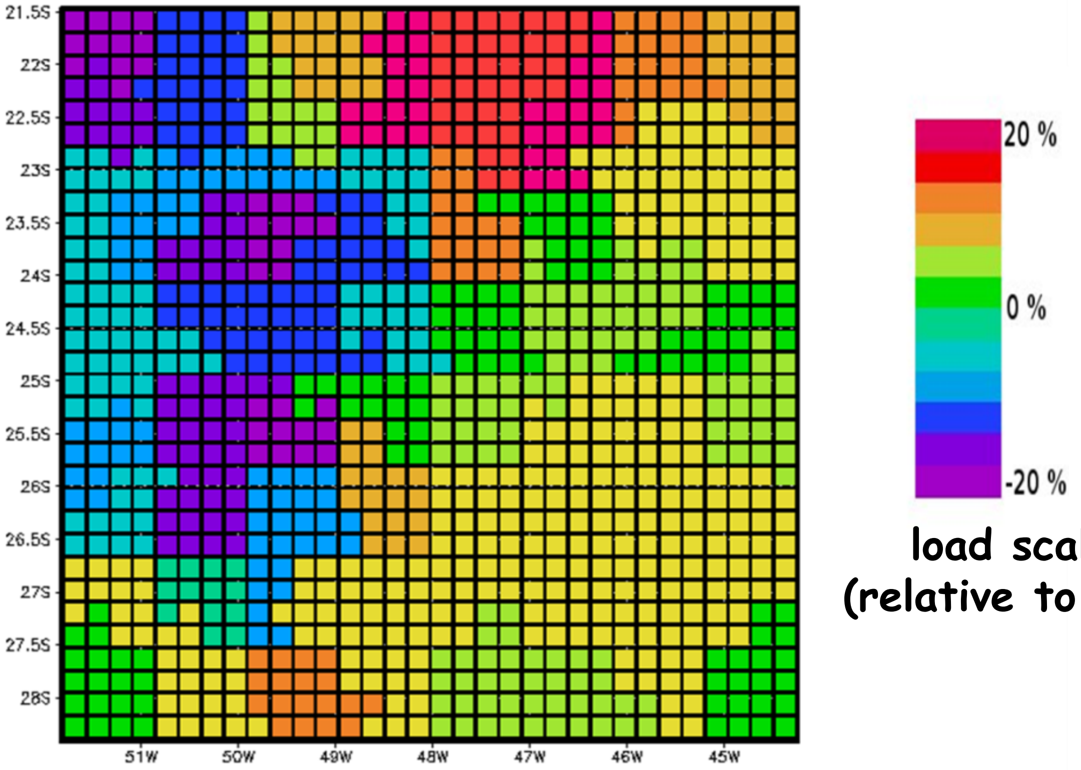
- Load distribution after Load Balance
  - Hilbert-LB balancer used (good for 2D domains)



load scale
(relative to average)

AMPI’s Load Balance in BRAMS

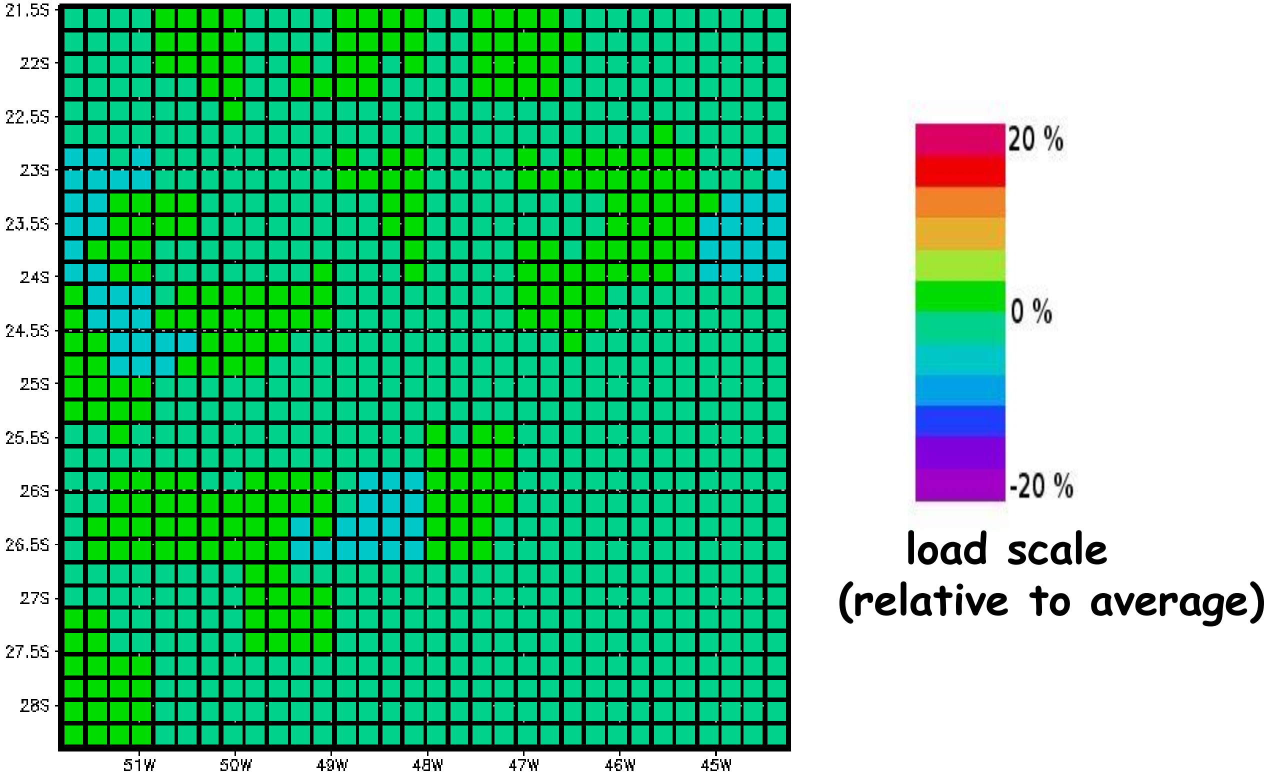
- Load distribution at 1200 tsteps
  - New imbalances arise!



load scale
(relative to average)

AMPI's Load Balance in BRAMS

- Load distribution after <u>second</u> LB
  - Applied at timestep 1200



AMPI's Load Balance in BRAMS

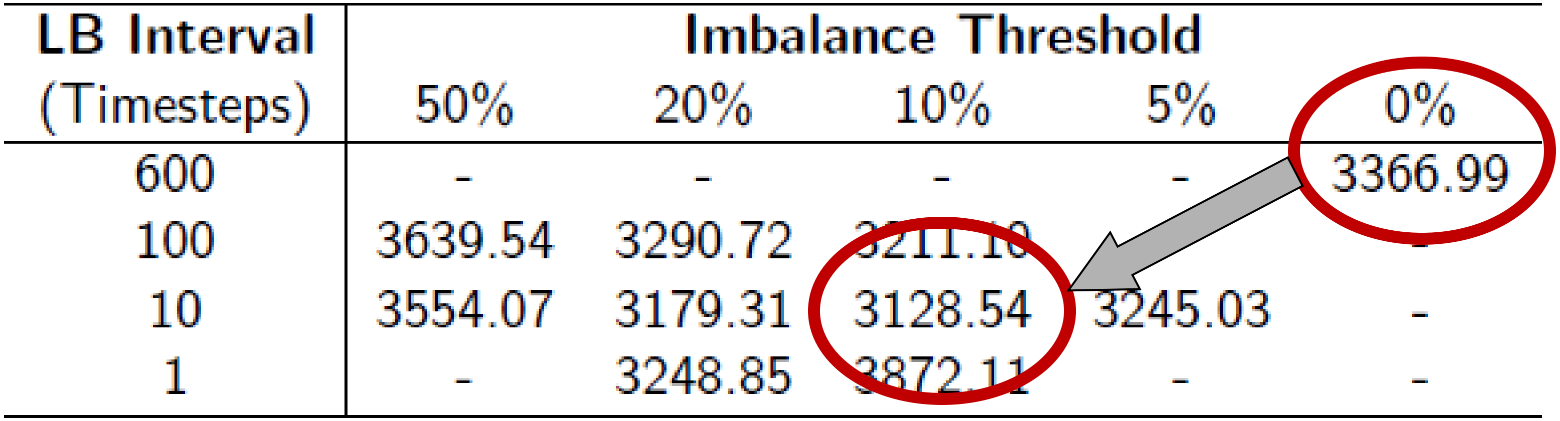
- Net Gain from Load Balancing

Configuration	Execution Time (s)
64 AMPI threads (no virtualization)	4,970
1024 AMPI threads	3,713 (-25%)
1024 AMPI threads + Load Balancing	3,367 (-32%)

- Question: when to rebalance?
  - Migrations are not free, even on SeaStar…
  - Rebalancing too often may be expensive (due to migration overhead)
  - Not rebalancing may lead to extended timestep durations (due to imbalances)
  - Imbalance sources change between forecasts!

AMPI's Load Balance in BRAMS

- Alternatives for rebalancing
  - Test various executions, distinct rebalancing periods
  - Activate migrations only when imbalance is "high" (i.e. greater than a given threshold)
  - Many tests conducted, with different parameters:



LB Interval	Imbalance Threshold				
(Timesteps)	50%	20%	10%	5%	0%
600	-	-	-	-	3366.99
100	3639.54	3290.72	3211.10	[illegible]	-
10	3554.07	3179.31	3128.54	3245.03	-
1	-	3248.85	3872.11	-	-

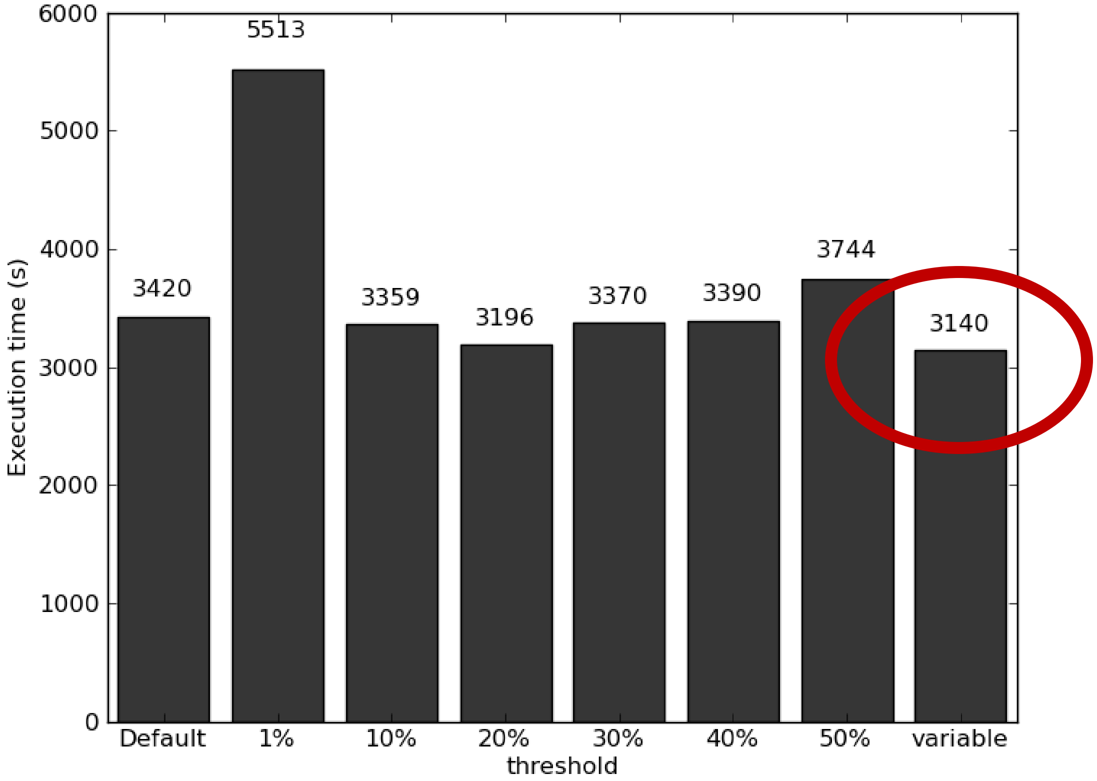
  - Can we find this sweet-spot automatically???

Automatic Adaptive LB Scheme

- Based on *Principle of Persistence*
  - Recent past may be a good estimator of near future
- Key Ideas
  - Observe recent time-step durations and loads
  - At rebalancing-evaluation moment: estimate benefits of rebalancing, considering migration costs
  - If profitable, allow migrations to occur
  - Call rebalancing evaluator frequently (cheap)
- Advantages
  - Relieves user from picking a threshold *a priori*
  - Should work for multiple (distinct) forecasts

Automatic Adaptive LB Scheme

- Evaluation of New Scheme
  - Threshold “selected” at rebalancing points (100 ts)
  - Forecast for RJ-2011 used, 5 Km resolution



Ongoing Work

- Focus: Systems with Accelerators
  - Effort-1: currently being done in Brazil (Panetta et al)
    a) Port key parts of the code to GPUs
    b) Continue to remove communication/network overheads
  - Effort-2: current visit of Prof.Alvaro Fazenda to Illinois
    - Use of AMPI to balance load in accelerated executions with GPUs
- Challenges (and Opportunities!):
  - Porting to accelerators in a portable fashion
    - Allow using both GPU and other accelerators (e.g. Intel-MIC)
  - Effective load balancing with accelerators
    - Characterization of load may vary in accelerators
    - Potential for use of CPUs and accelerators concurrently
      - Truly heterogeneous load balancers are needed

Conclusion

- Adaptive Load Balancing
  - Can be done via AMPI with low programmer effort
  - Allows addressing dynamically-arising imbalances
  - Enables smarter, automatic LB schemes in BRAMS
- Research Challenges
  - Exploiting accelerators in a portable manner
  - New balancing policies for accelerators
  - Advanced load balancers for heterogeneous systems

Acknowledgments

- U.S.Dep.Energy: DE-SC0001845 (HPC-Colony-2)
- U.S.Nat.Science Foundation OCI-0725070 (Blue Waters)
- CAPES agency – government of Brazil
- TeraGrid allocation ASC050039N