RAMS - BASED SOFTWARE DEVELOPMENT HISTORY AT CPTEC

Jairo Panetta

May 2006

Background:
Software Development at CPTEC

HPC Group at CPTEC: Project Driven by HPC Speciality

Alvaro Luiz Fazenda	PhD	Project	MPI, Vet	CATT
Daniel Merli Lamosa	MsC	Project	Proc, Linux, Tools	BRAMS
Eduardo Hidenori Enari	PhD	Project	Exper, Tools	BRAMS
Jairo Panetta	PhD	Project	All	All
José Paulo Bonatti	PhD	INPE	Modelagem	GLOBAL
Luiz Flavio Rodrigues	Grad	INPE	OMP, Vet	CATT, ETA
Marcio Augusto Moraes	PhD	Project	MPI, Vet	PSAS
Paulo Kubota	MsC	FUNCATE	Acop, Vet	GLOBAL, CATT
Pedro L. S. Dias	PhD	Conv. IAG	Modelagem	GLOBAL, BRAMS
Roberto Pinto Souto	PhD	Project	Grid	BRAMS
Simone Shizue Tomita	MsC	FUNCATE	OMP, Vet	GLOBAL, ETA
Saulo Barros	PhD	Project	MPI, Mod, Vet	GLOBAL

PART 1:
History of RAMS – based projects

BRAMS (1999 – 2003)

- ELEBRA – IAG – ASTER (Finep)
- CPTEC – IAG – ASTER (Finep)
  - Production Model for Regional Weather Centers; CPTEC should do sw maintenance
  - Aimed towards clusters of PCs
  - Restricted distribution (Proprietary)
  - Uses RAMS 5.0

BRAMS Project Goals

- Increase forecast quality for the tropics
  - Shallow Cumulus + SIB + Soil Humidity Initialization + Tropical Vegetation
- Better Codification
  - Implicit None + Replacing Commons by Modules + Types + Binary Reproducibility
- Faster sequential and parallel execution
  - Sequential gain on Diffusion (RAMSIN dependent)
- Research and Production Versions
  - BRAMS 1.0 (over RAMS 5.0.2)
  - BRAMS 2.0 (over RAMS 5.0.4)
  - BRAMS 3.0 ….

BRAMS Difficulties

- Flexibility is central (due to research)
- Keep as many methods in the code as possible
  - User-selection Convection, Radiation, Microphysics, etc..
- Use input switches to select methods at run time
  - On some cases, multiple switches for a method
  - Just too many input switches combinations
- As a result:
  - Very flexible code, but
  - Some switch combinations do not work…(array bounds, initialization, inconsistencies)
  - Impossible to optimize and to guarantee correctness
    - Reproducibility redone 4 times
    - Keep finding bugs as of today…

CATT-BRAMS (2004 - today)

- Environmental model for CPTEC
- CATT-BRAMS = BRAMS +
  - CARMA Radiation Module
  - Biomass Burning Module
  - Pollution Diffusion Module + ...
- Have to move from PC clusters to NEC SX-6
- As a result:
  - Vectorizaton becomes a central issue
    - 150 MFlops/Processor on early 2004; 600 MFlops/Processor on late 2005
    - References: Global model (3500 MFlops/Processor); ETA (1200 MFlops/Processor)
  - Parallel Efficiency becomes a central issue
    - Shared Memory parallelism not exploited
    - Dynamic Load Balancing is badly needed
  - Bugs are a nightmare
    - SX-6 is very peaky at run-time

BRAMSNET (2005 - 2006)

- CPTEC – IAG – UFCG – UFRJ – FURG (Finep)
- Daily production feedbacks research and maintenance
- BRAMS becomes Free Software
- Web site (www.cptec.inpe.br/brams) for sw distribution, documentation, input data and discussion list
- As a result:
  - Increases user's pressure for documentation, software quality and efficiency
  - BRAMS execution time is quoted against other models

GBRAMS (2005 - 2006)

- CPTEC – LAC/INPE – UFRGS (Finep)
- 50 years climatology over South America
- 3 clusters spread over Brazil, accessible by a portal
  - Year-long runs; restart by history file
  - 3 regions of Brazil
  - 3 starting dates
- Portal tests three grid middleware schemes over time
- As a result:
  - Binary reproducibility with (or without) history files is central
  - More bugs …

SegHidro (2005)

- UFCG - CPTEC – FUNCEME (Finep)
- Portal to cascade:
  - 2 atmospheric models (BRAMS and RSM)
  - Hydrological models (grid enabled)
  - Risk Analysis models (grid enabled)
- Used by state authorities (water usage)
- BRAMS extensively used for weather and climate forecast, assimilating CPTEC
- As a result, extensive use and test of BRAMS

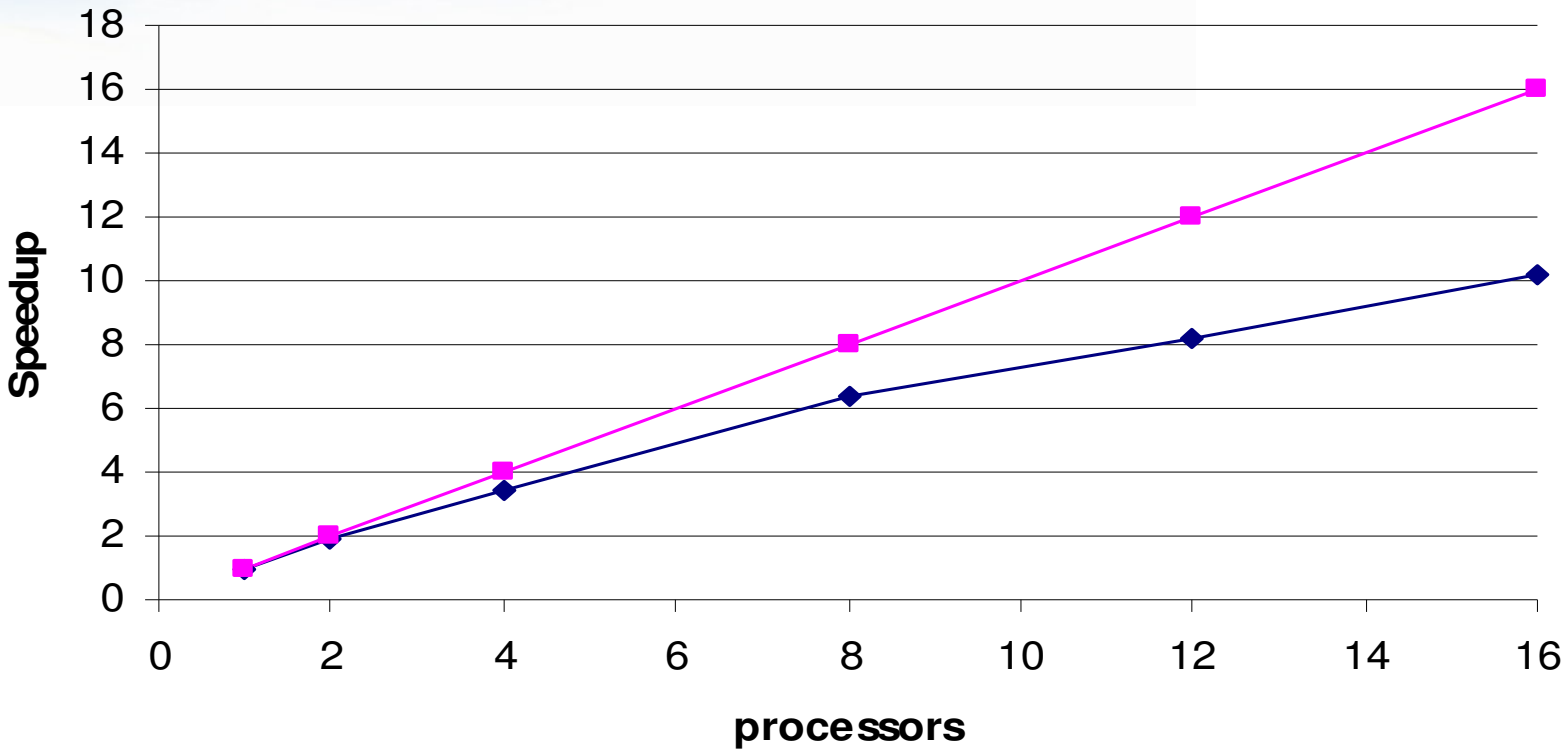
Summary

- BRAMS (1999 – 2003)
  - Own modules + better codification
- CATT-BRAMS (2004 – today)
  - CPTEC environmental model; requires SX-6 efficiency
- BRAMSNET (2005 – 2006)
  - BRAMS dissemination; user feedback; pressure on PC Cluster efficiency and documentation; quoting against other models
- GBRAMS (2005 – 2006)
  - Grid Climatology; reproducibility by history file
- SegHidro (2005)
  - Extensive use and test; quoting against other models

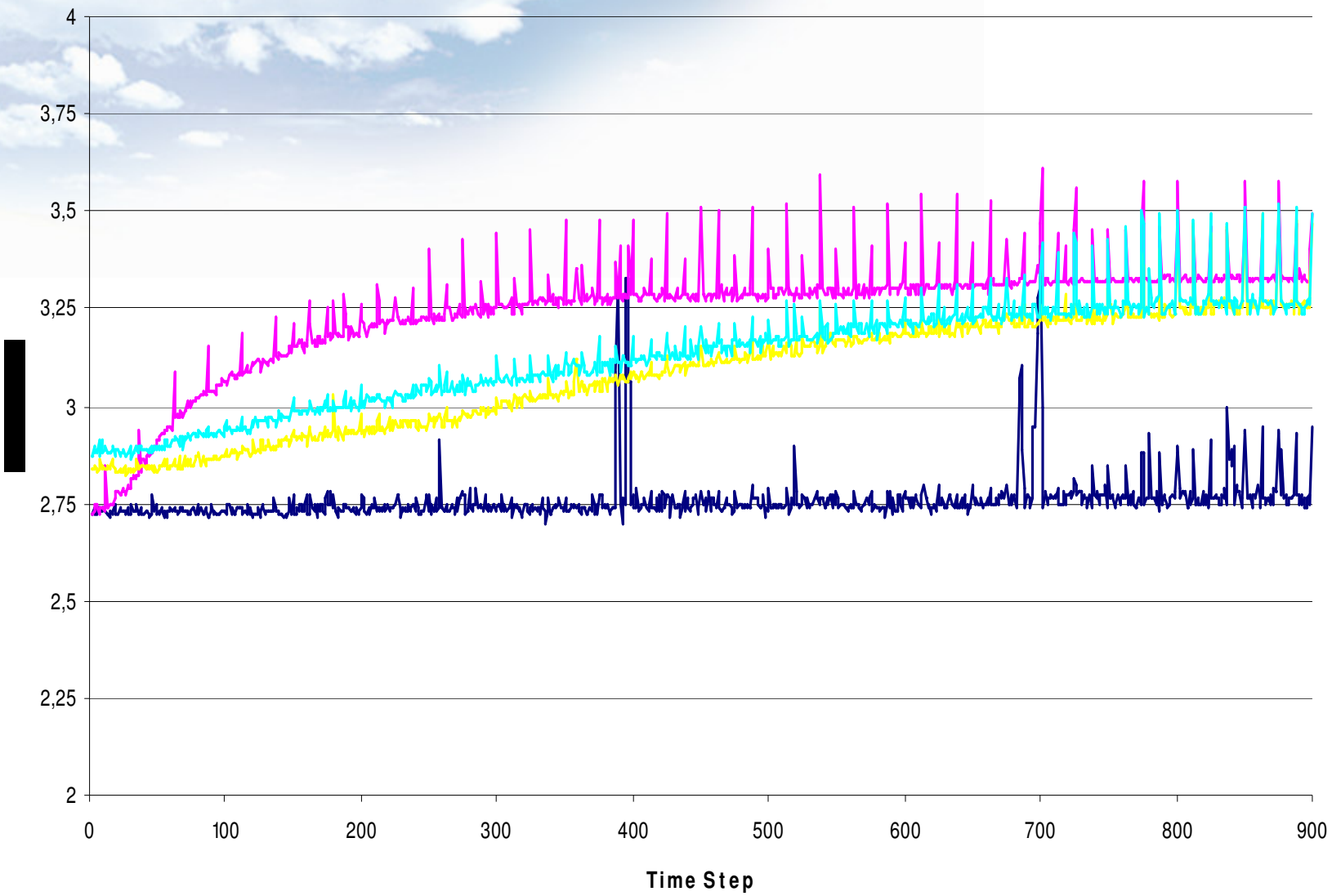
PART 2:
BRAMS Execution Time Experiments

The Need for Dynamic Load Balancing

Restricted Scalability

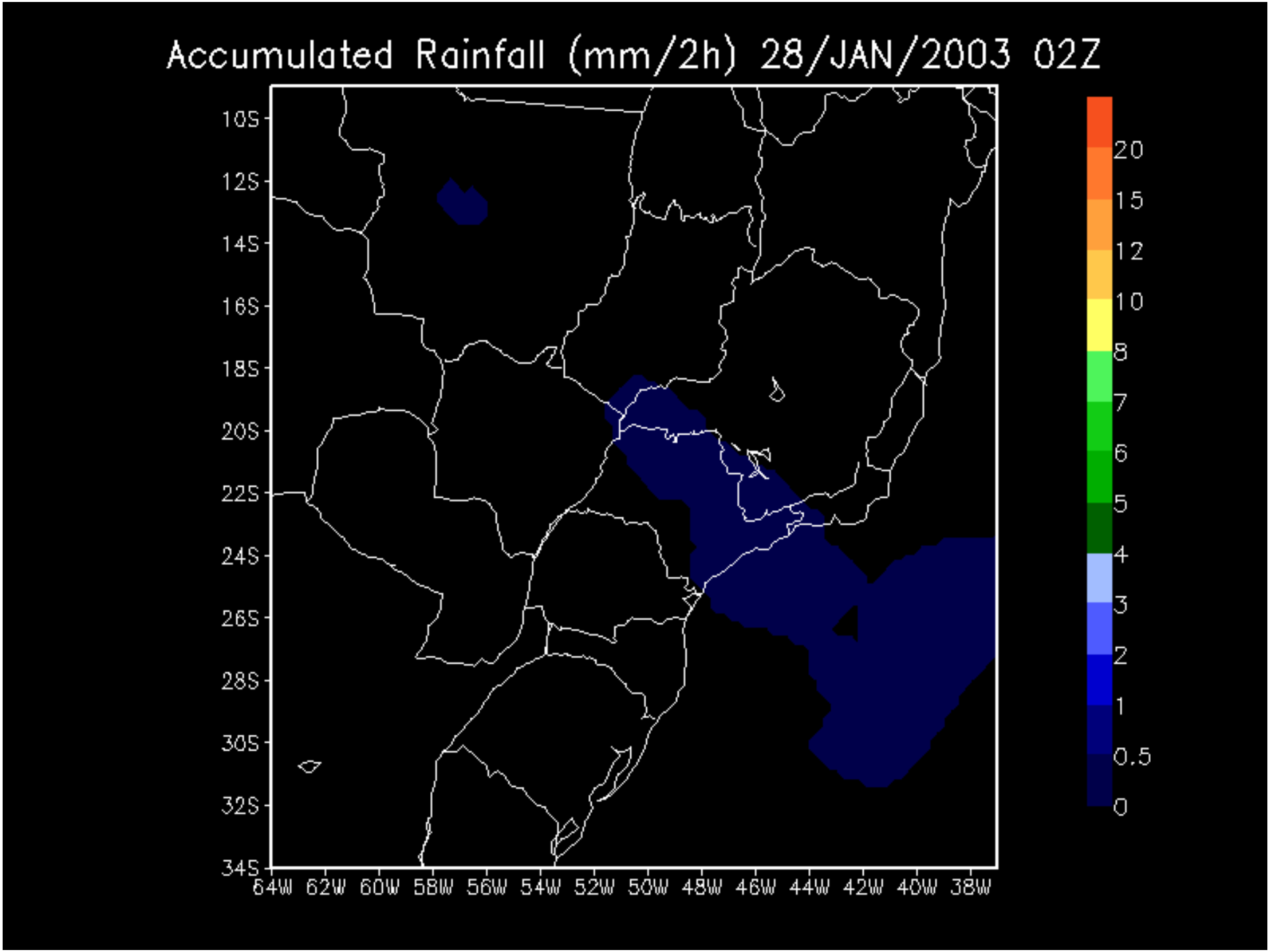


Execution time per Time Step

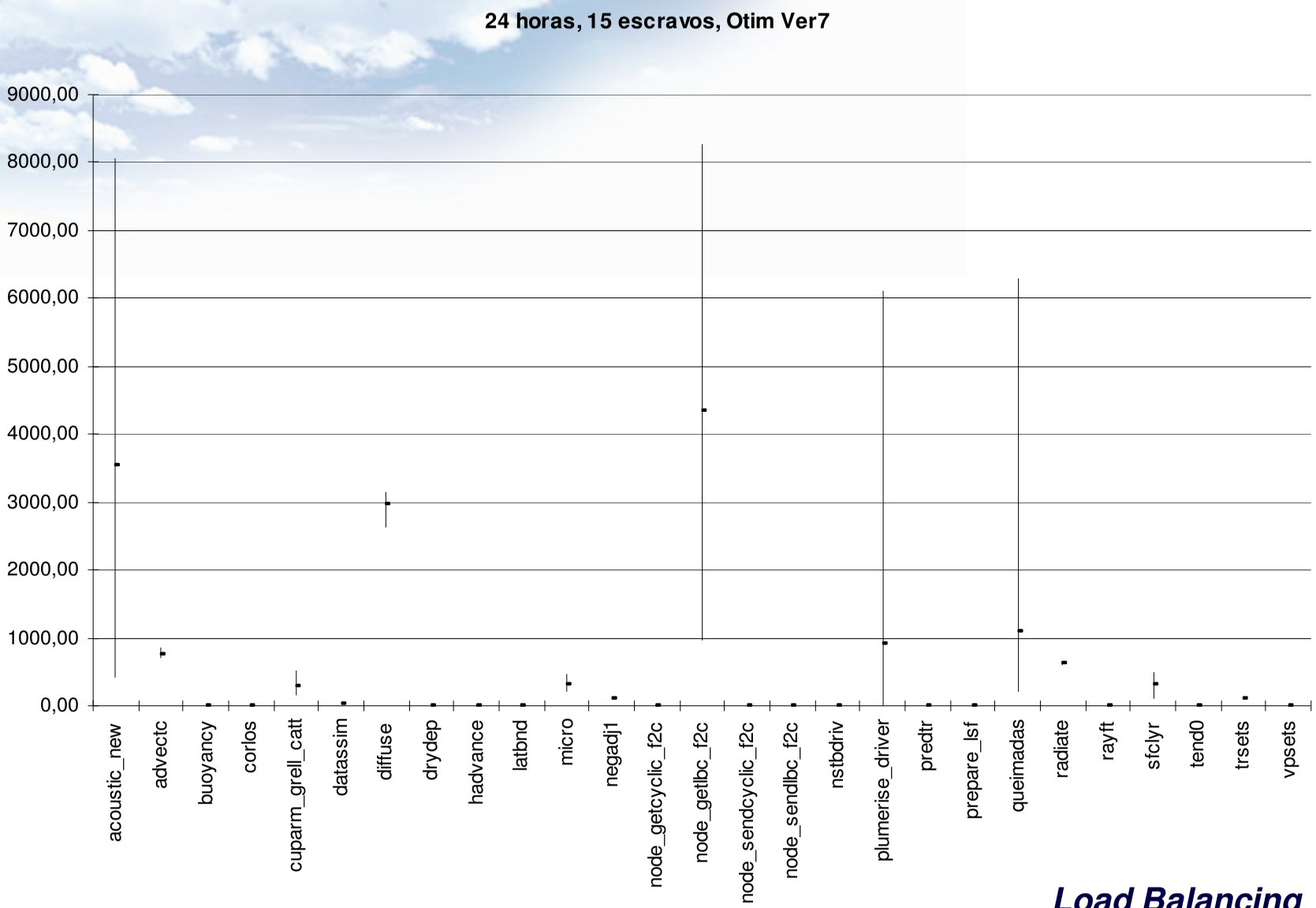


Load Balancing

Domain Decomposition

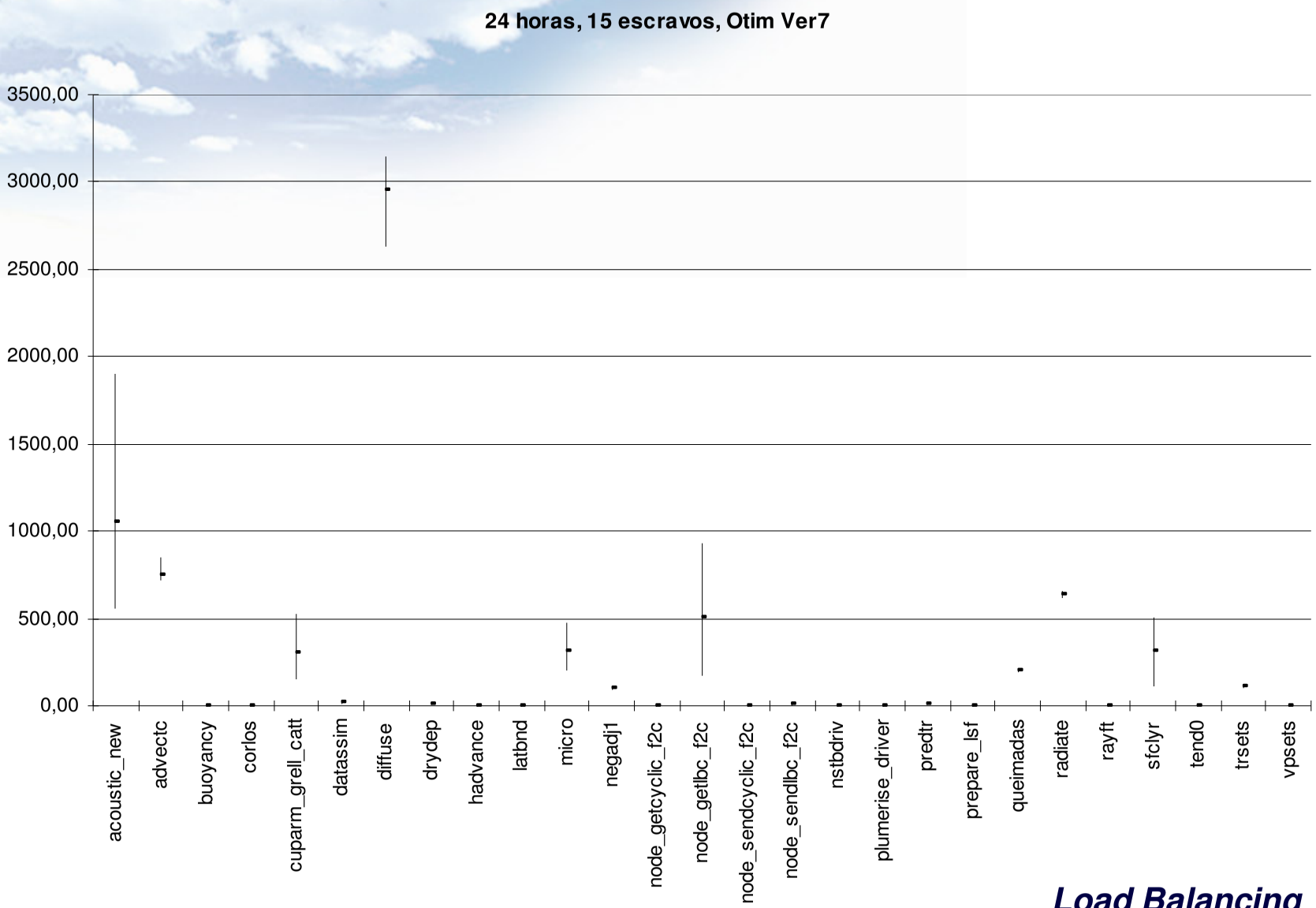

Accumulated Rainfall (mm/2h) 28/JAN/2003 02Z
10S
12S
14S
16S
18S
20S
22S
24S
26S
28S
30S
32S
34S
64W 62W 60W 58W 56W 54W 52W 50W 48W 46W 44W 42W 40W 38W
20
15
12
10
8
7
6
5
4
3
2
1
0.5
0

CATT-BRAMS on SX-6



Load Balancing

CATT-BRAMS on SX-6



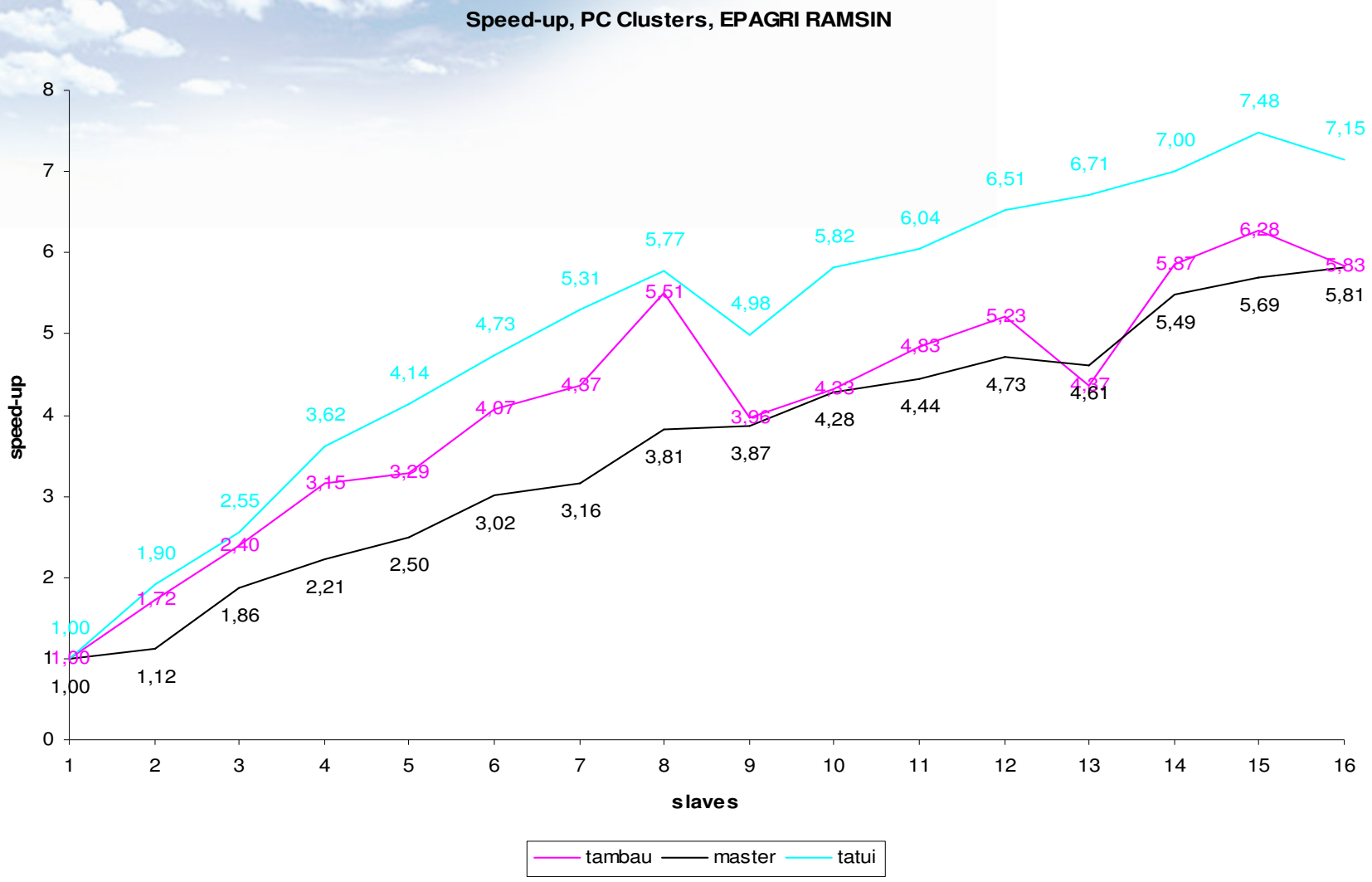
Load Balancing

Summary

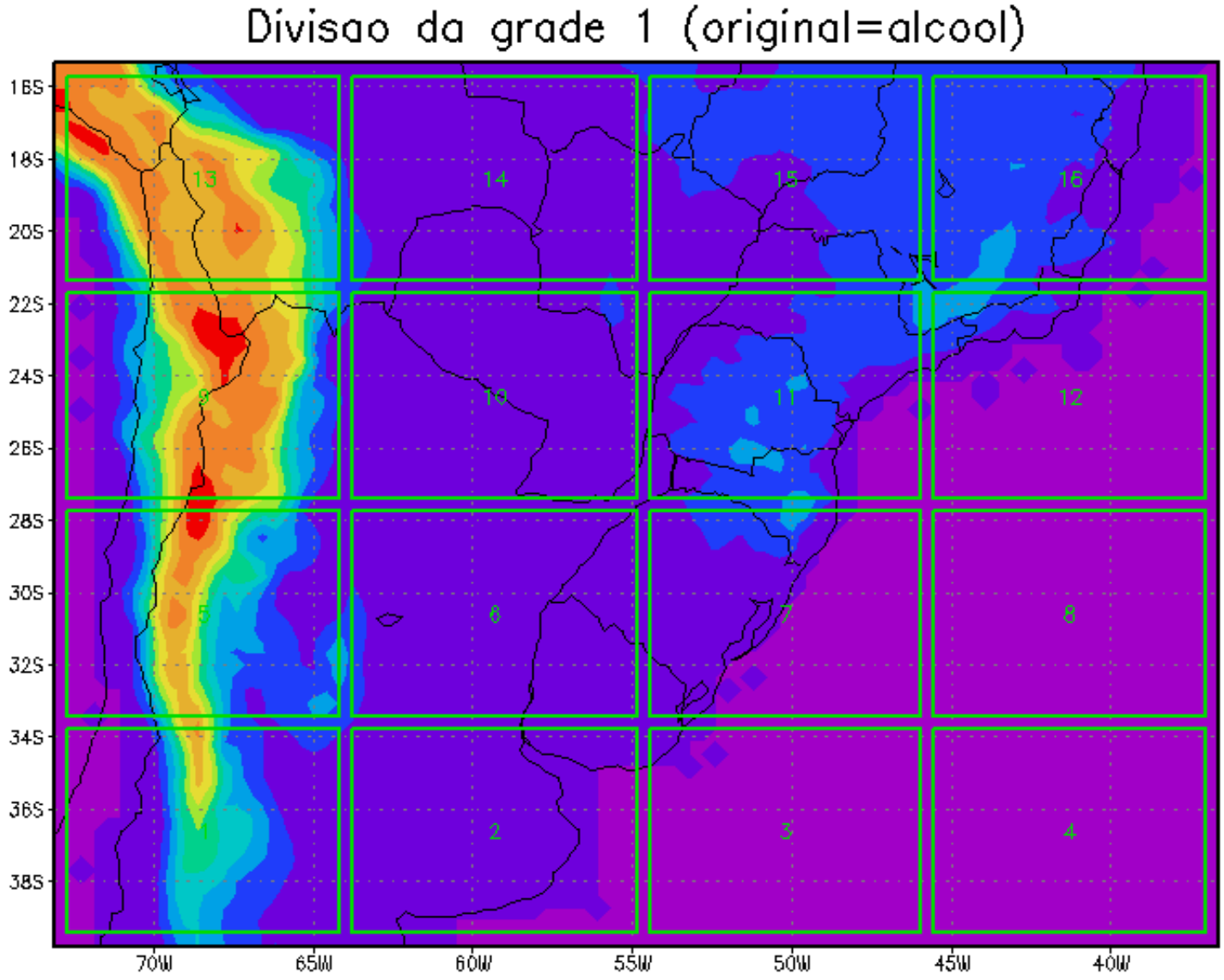
- Sources of load unbalancing:
  - Dynamically localized precipitation on purely atmospheric models;
  - Dynamically localized emission sources on environmental models;
- While precipitation and emission codes are computationally heavy, dynamic load balancing is badly needed

The Need for Sequential Optimization

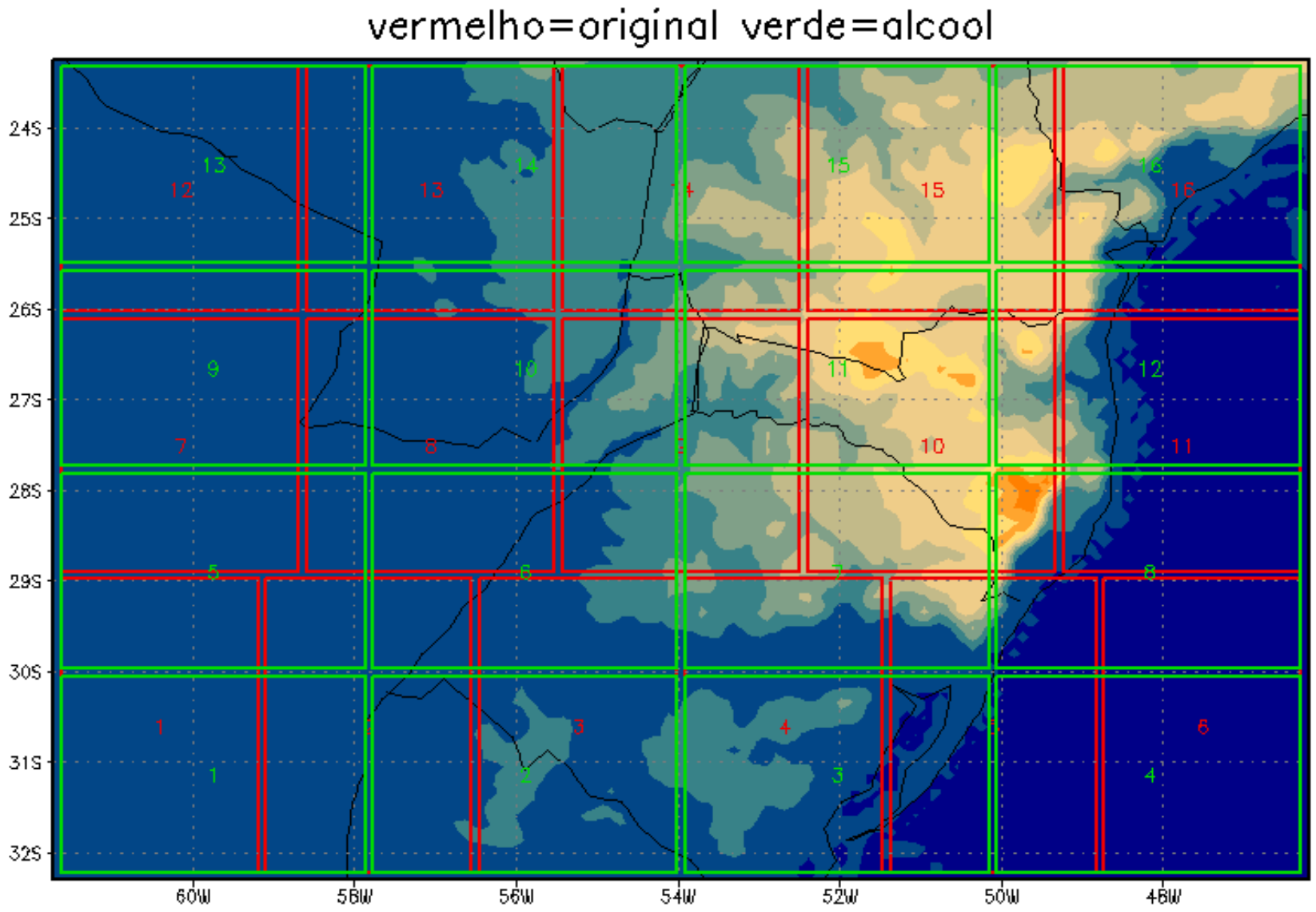
The impact of computing architecture



External Grid – 40 km resolution



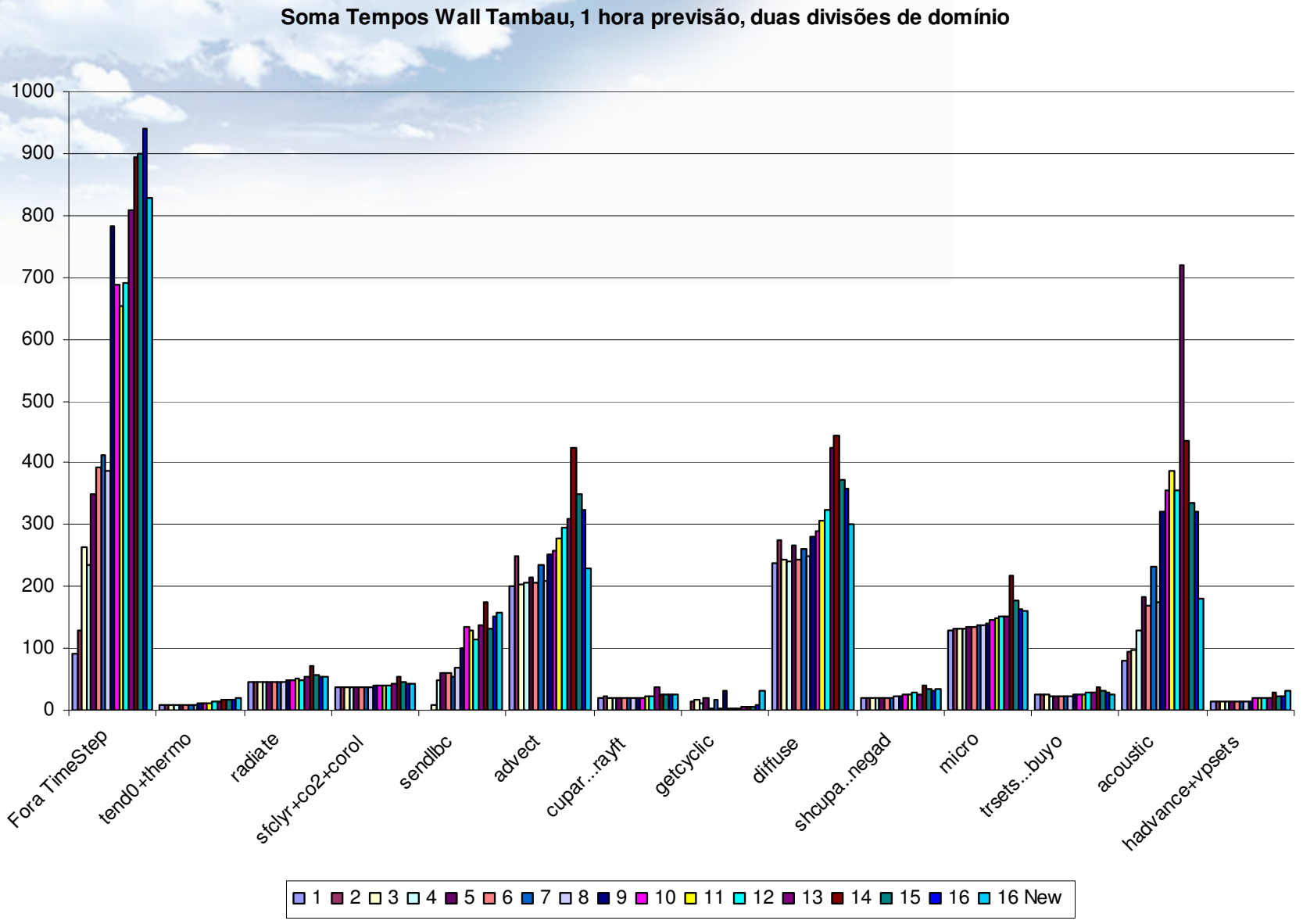
Coarser Gird – 10 km resolution



Specific Domain Decomposition

- By changing domain decomposition, execution time on 16 slaves was reduced by 25 % (average over 30 days)
  - From 4 hours 20 minutes
  - To 3 hours 14 minutes
- Why?
  - Partially due to lower wait on communication
  - Partially due to better processor usage

Wall Clock sum over processors

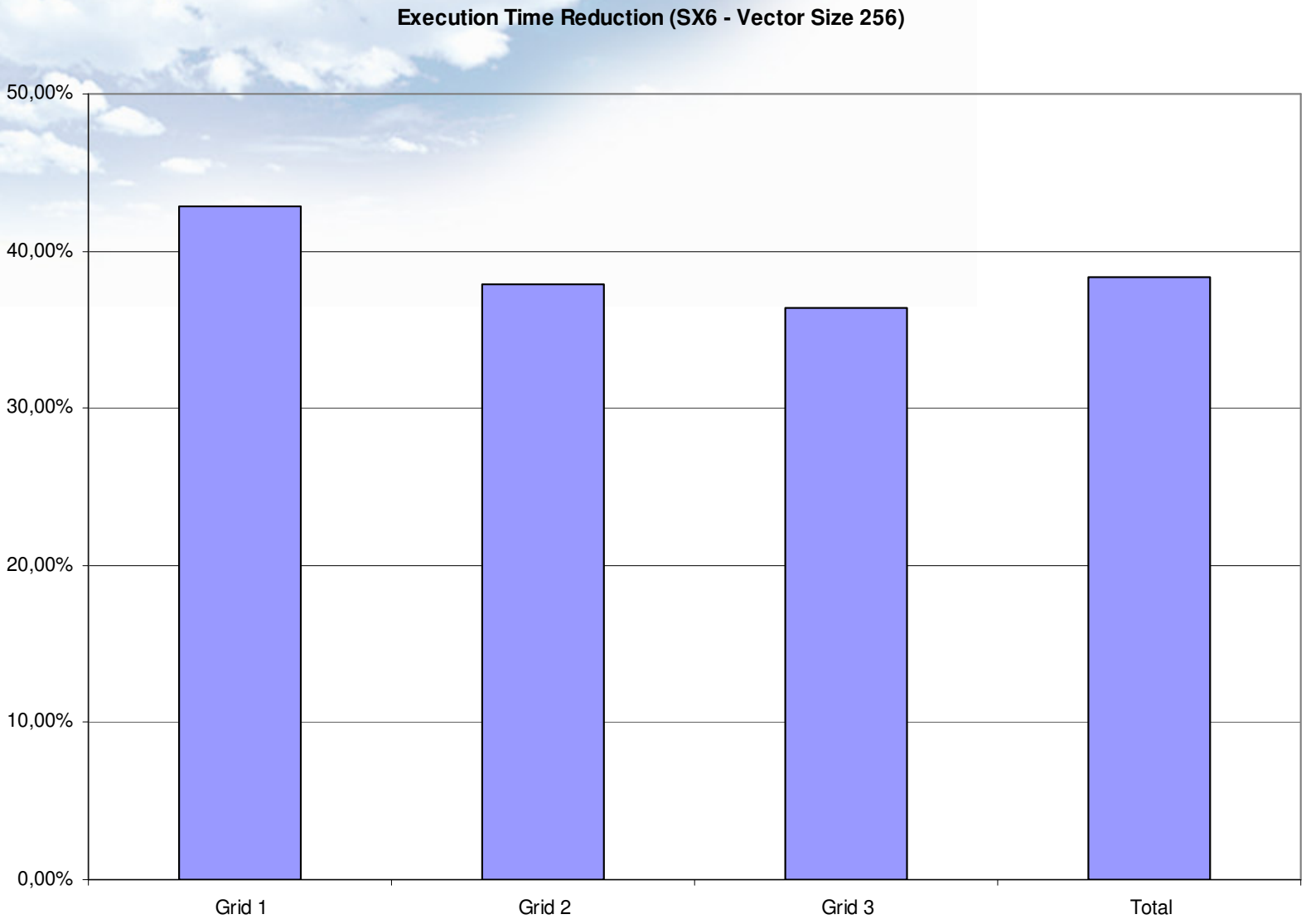


How to do Sequential Optimization

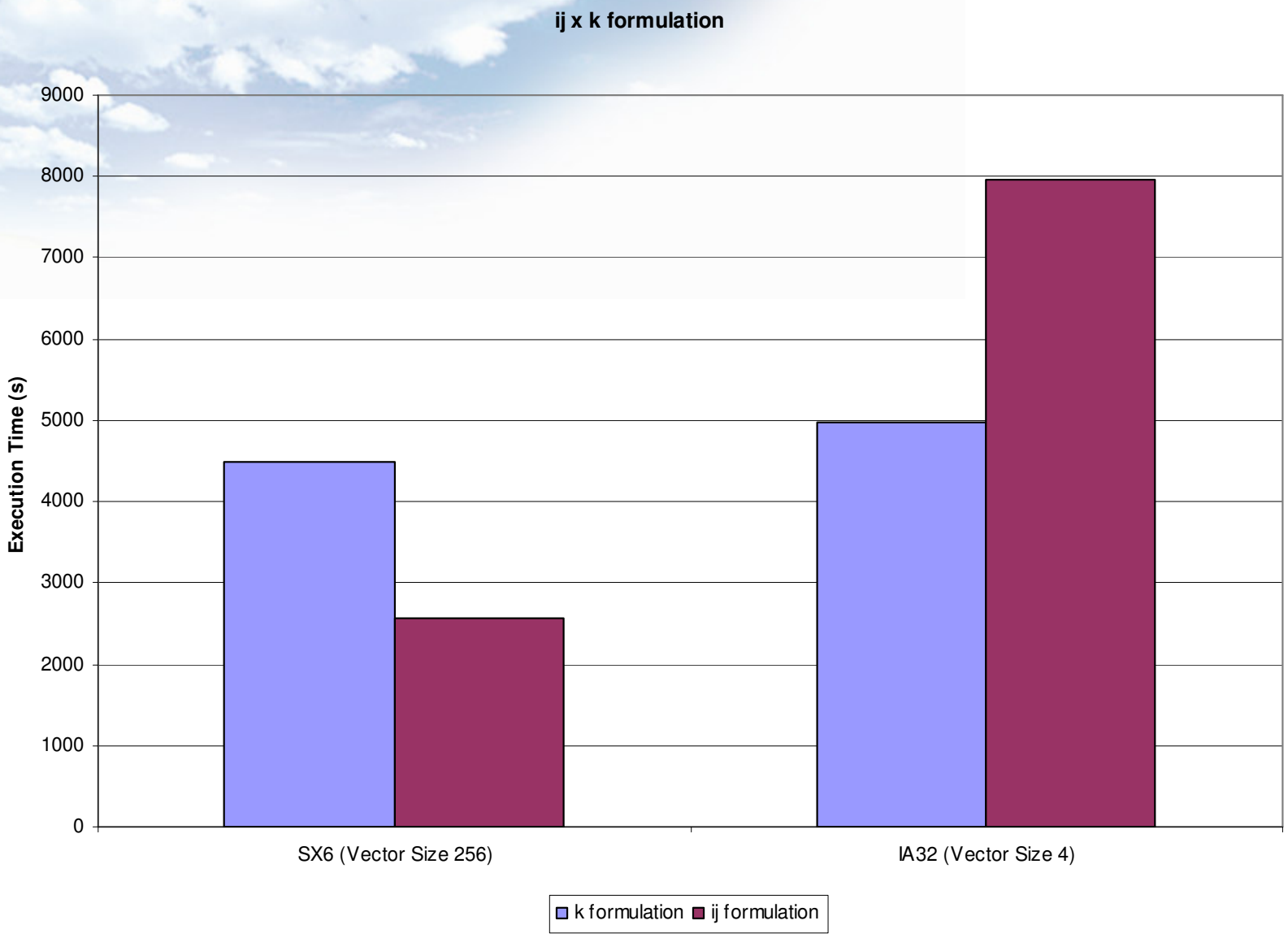
Vectorizing BRAMS - SX-6

- Inner loops over verticals (k)
  - Each physics module receives one atmospheric column
  - Driver visits each surface point (i,j), invoking module
- Due to that:
  - Small vectors (32 vertical layers)
  - Poor vectorization (dependencies on vertical)
  - SX-6 was 5% faster than PC
- Master Plan: Inner loops over surface points (ij)
  - “Unstructured” grid (any set of surface points)
  - First step: radiation and microphysics
  - 2 man x year effort

ij x k formulations (SX6)



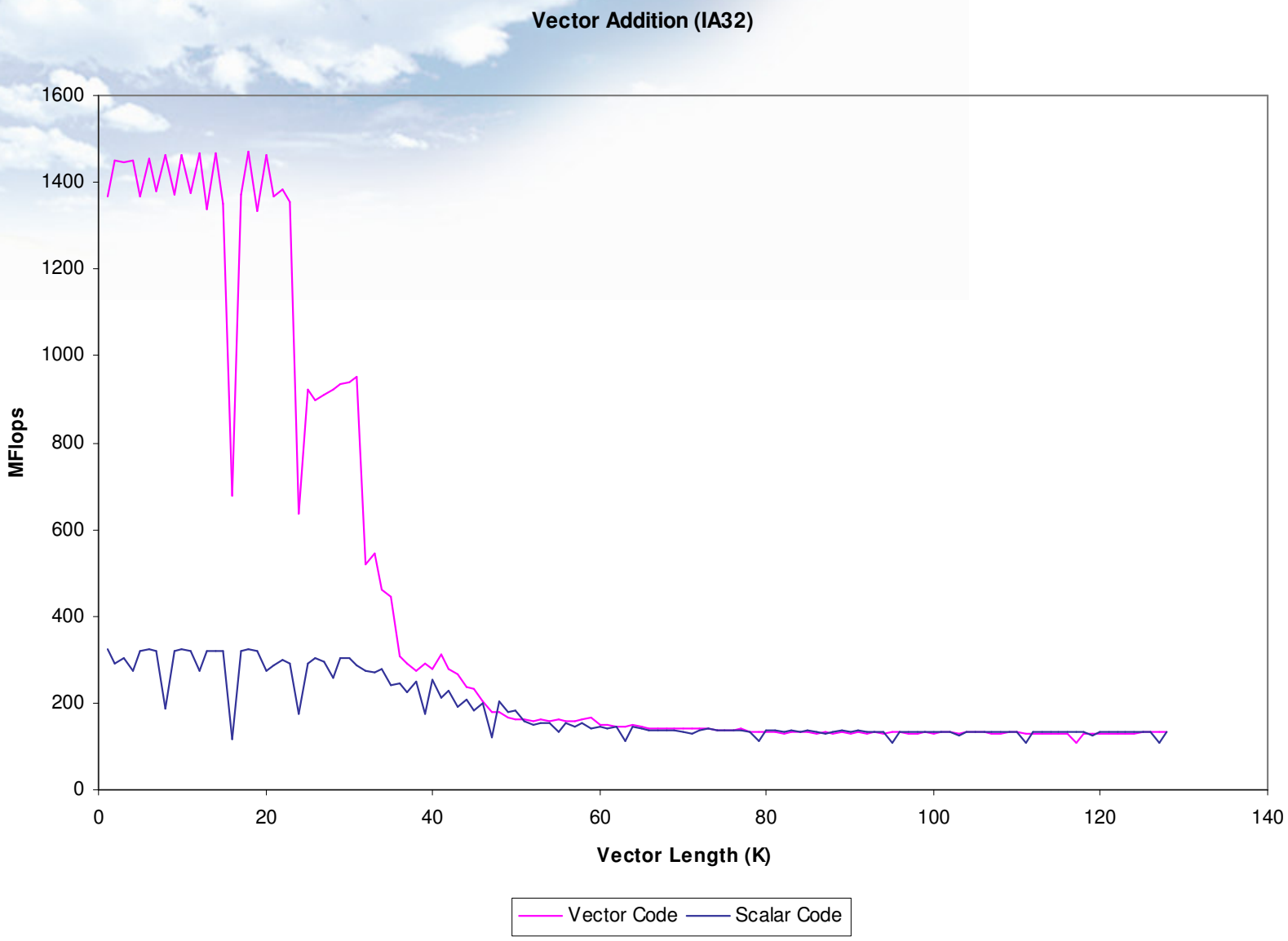
SX6 and IA-32


ij x k formulation
Execution Time (s)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
SX6 (Vector Size 256)
IA32 (Vector Size 4)
k formulation
ij formulation

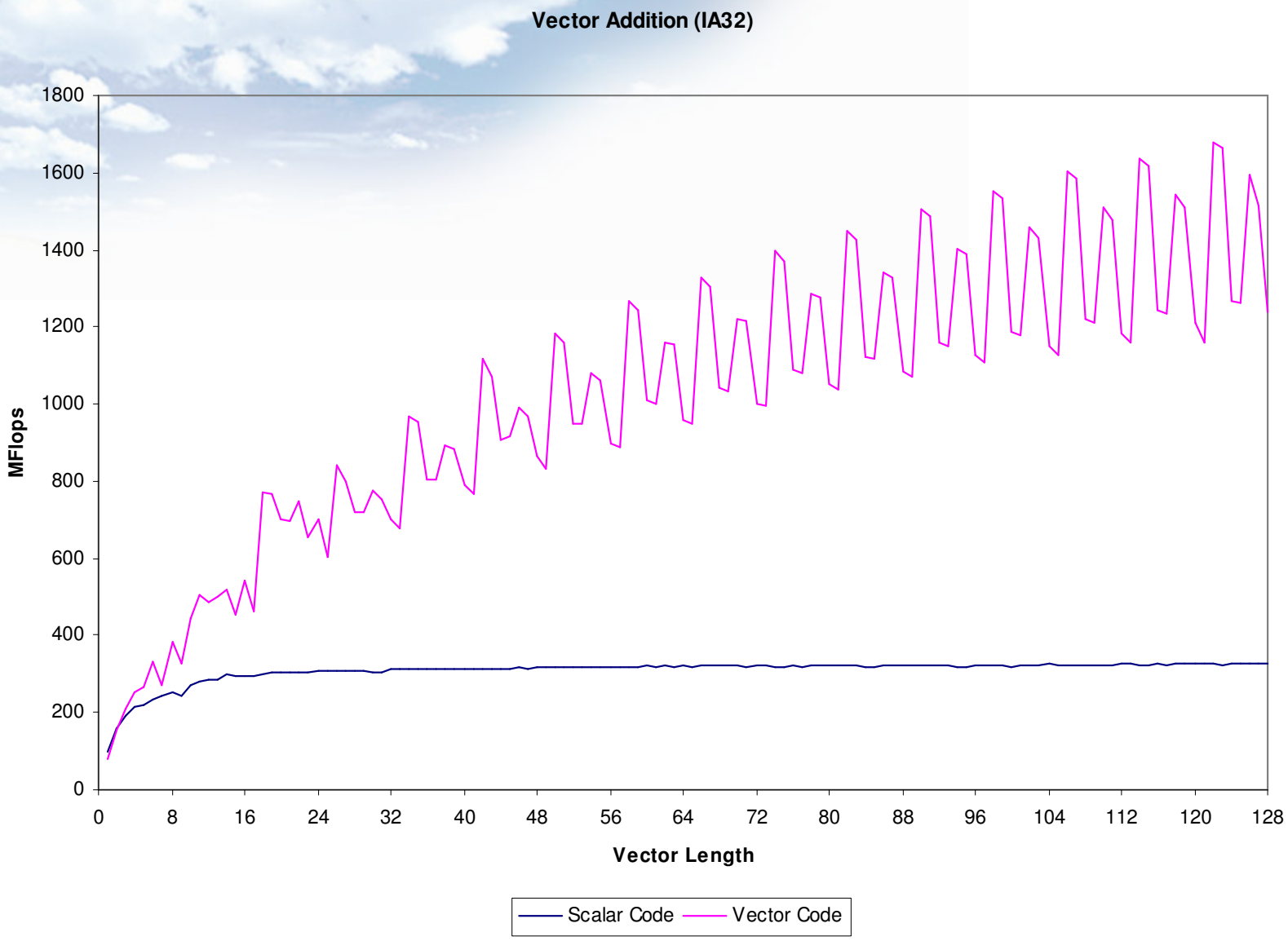
Vectorizing on IA32

- SSE Vector Instructions:
  - Vectors of fixed size (1 or 4)
  - No vector masked instructions
- Intel compiler:
  - Reasonable vectorization (scalar expansion)
  - Cannot vectorize loops with conditionals
  - Unroll loops with depth 8
- Few performance profilers:
  - Require specific kernels

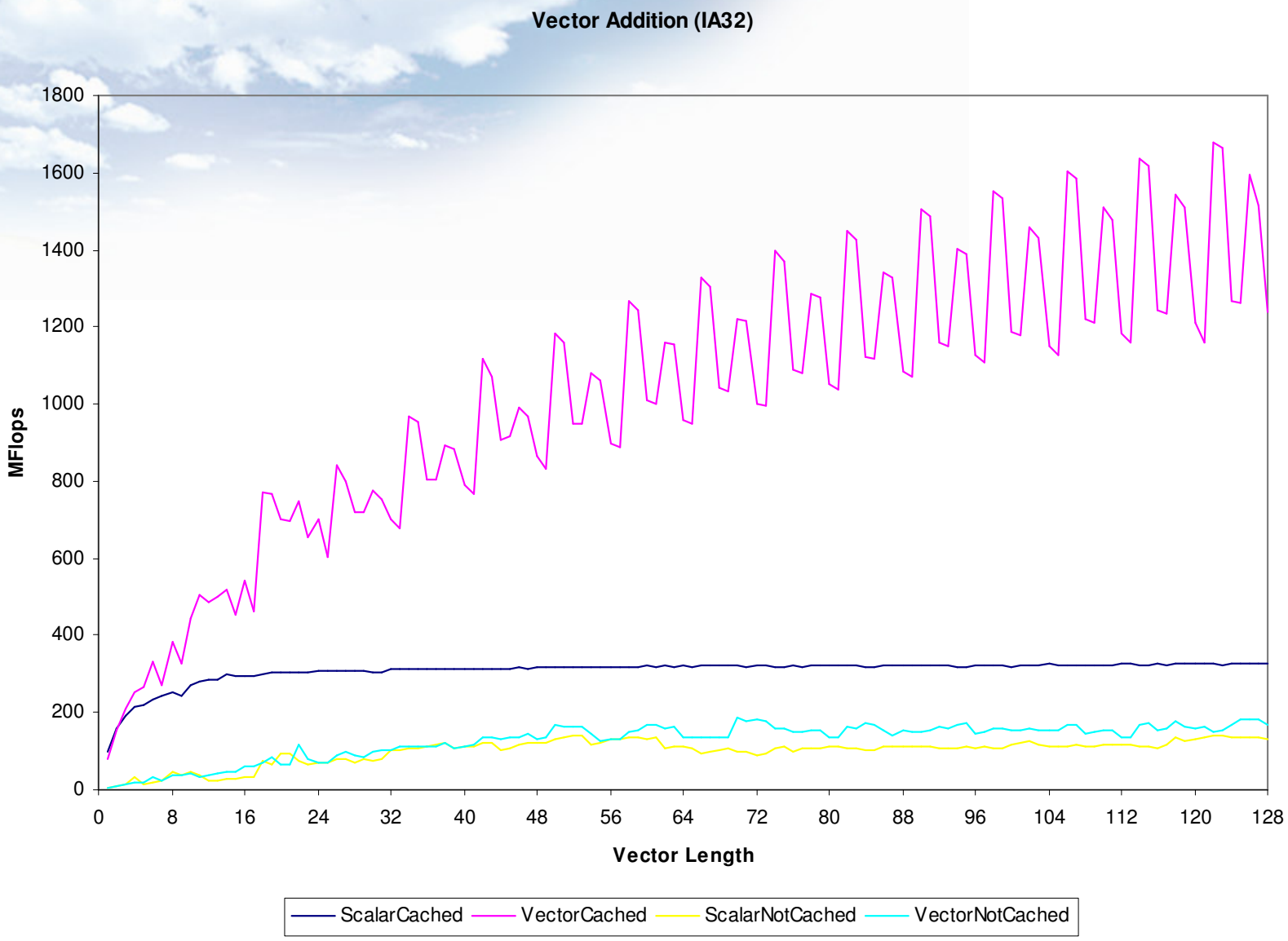
Vectorizing on IA 32



Vectorizing on IA 32



Vectorizing on IA 32

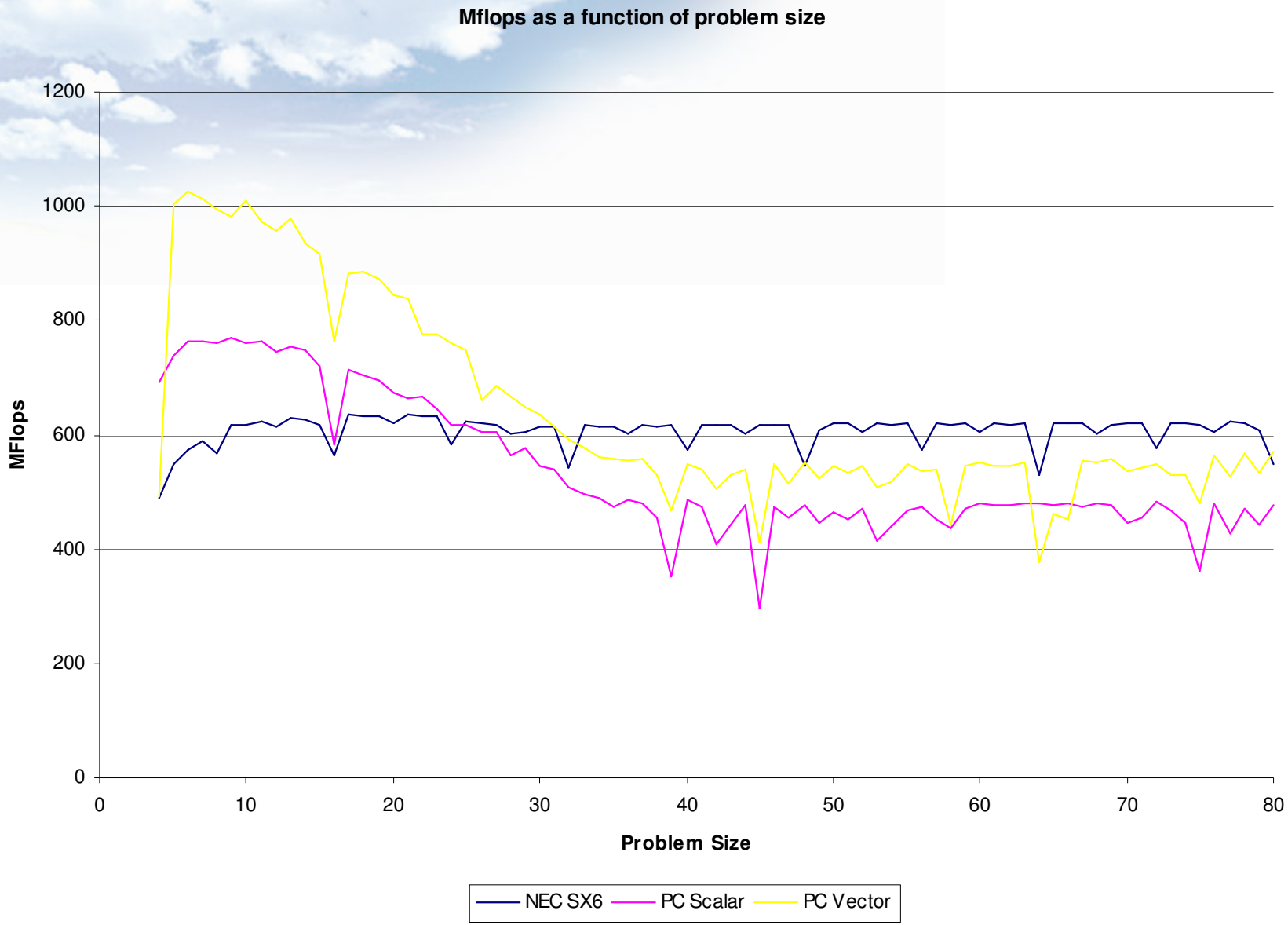


Vectorizing on IA 32

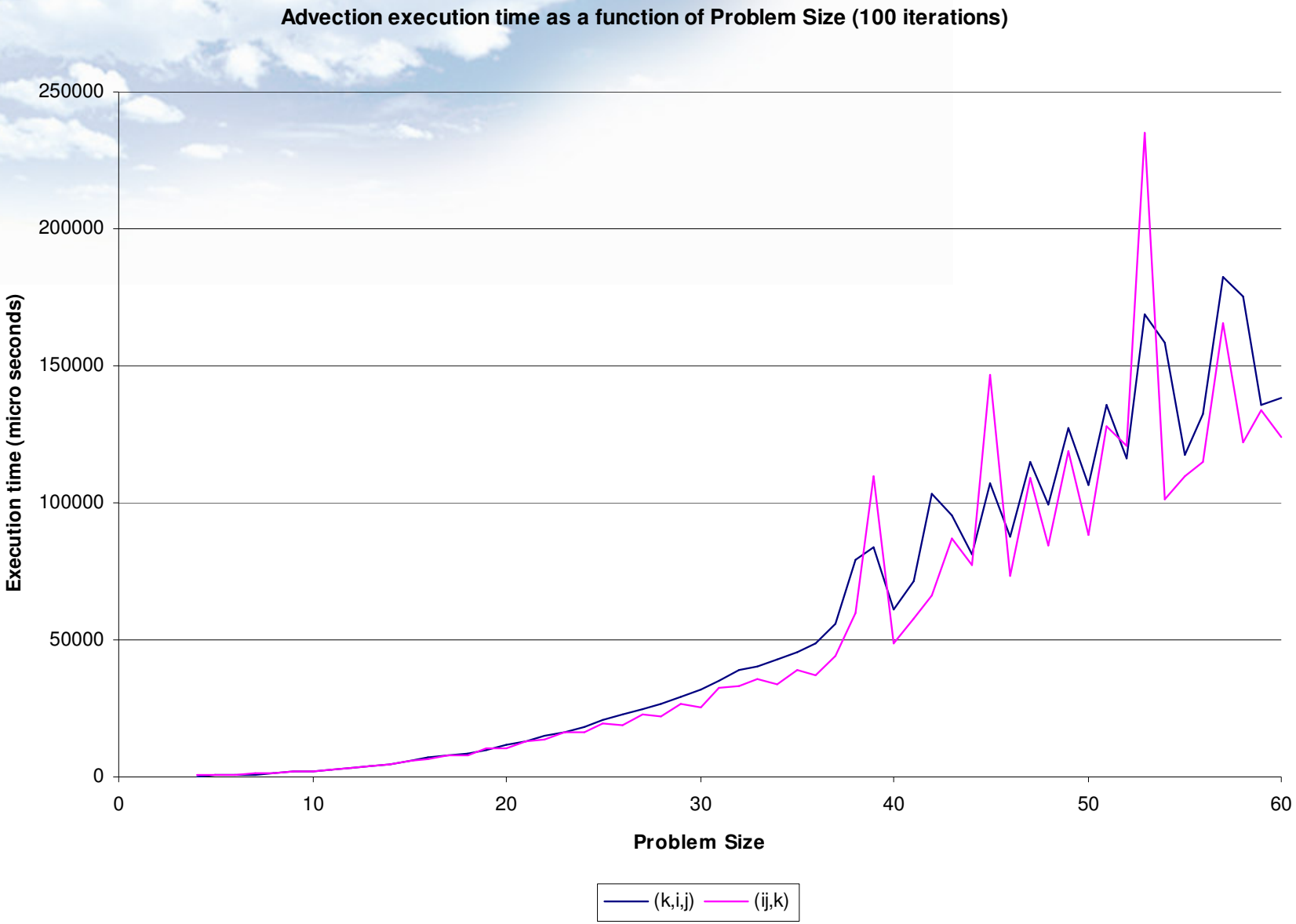
- Conclusions:
  - Potentially high gain in vectorization
  - Gain increase with vector length (modest size)
  - Cache re-use is central
  - Lack of vector mask inhibits vectorization gains on loops with conditionals

Case Study: Advection

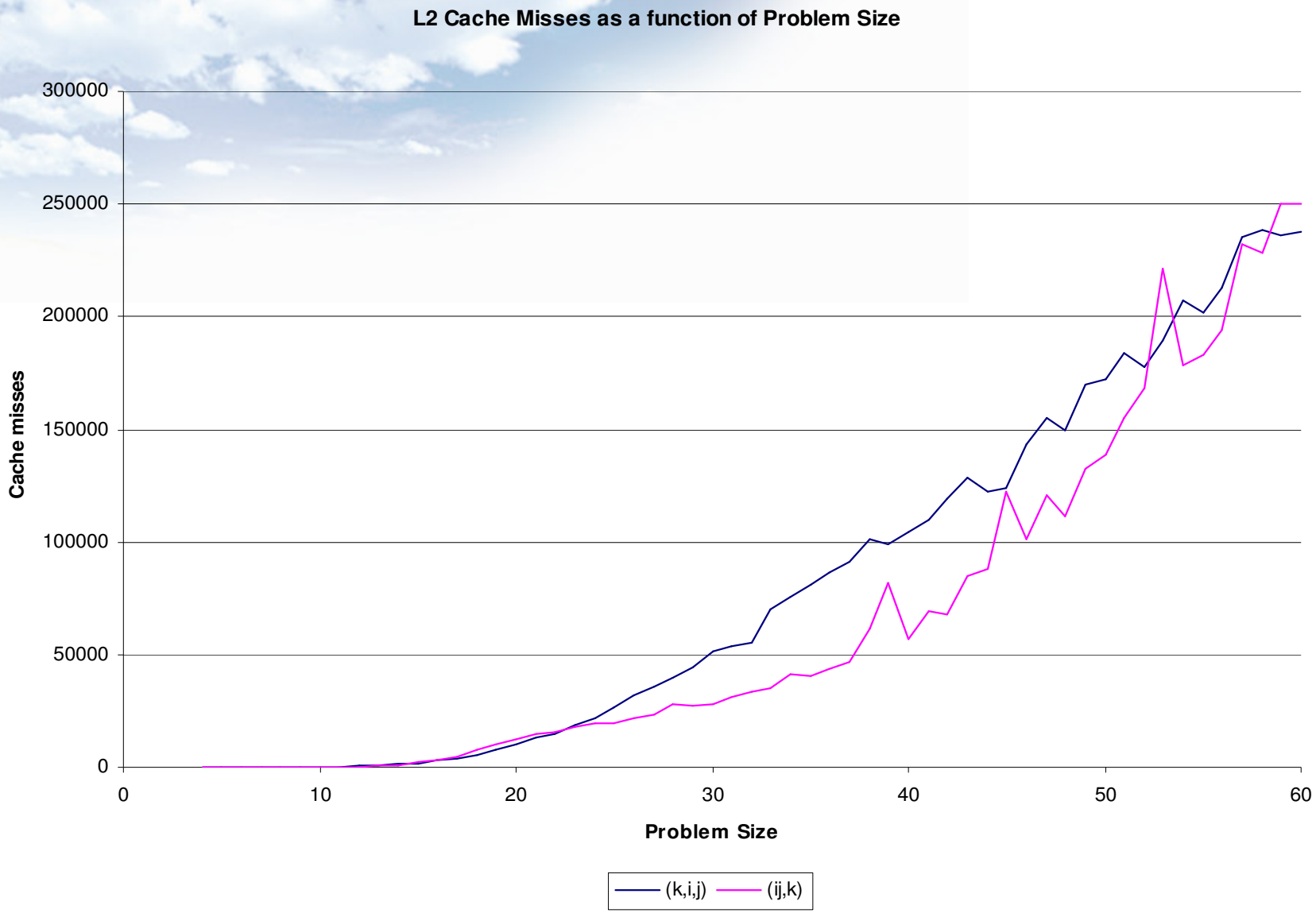
Original Performance (k)



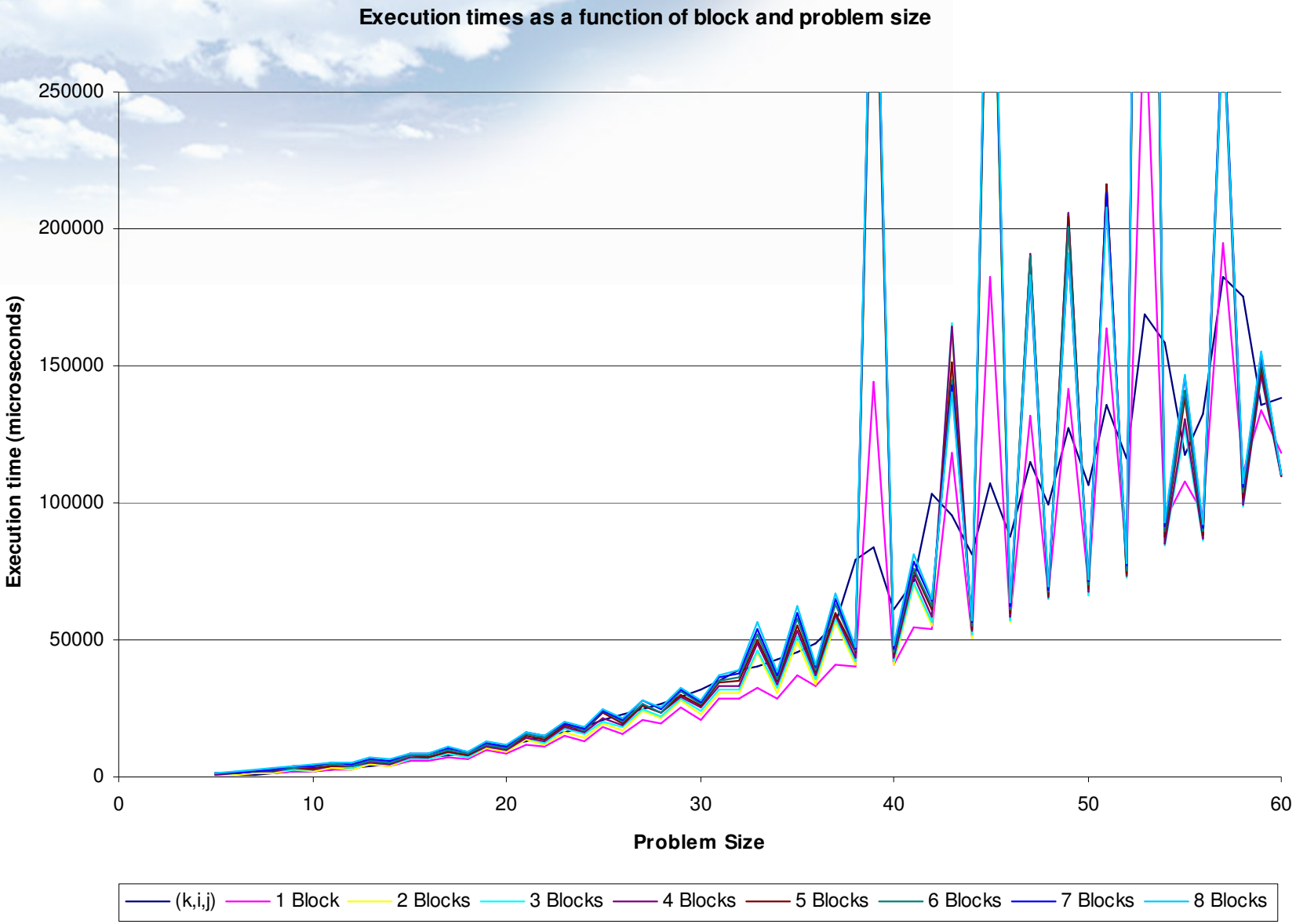
ij Formulation Version 1



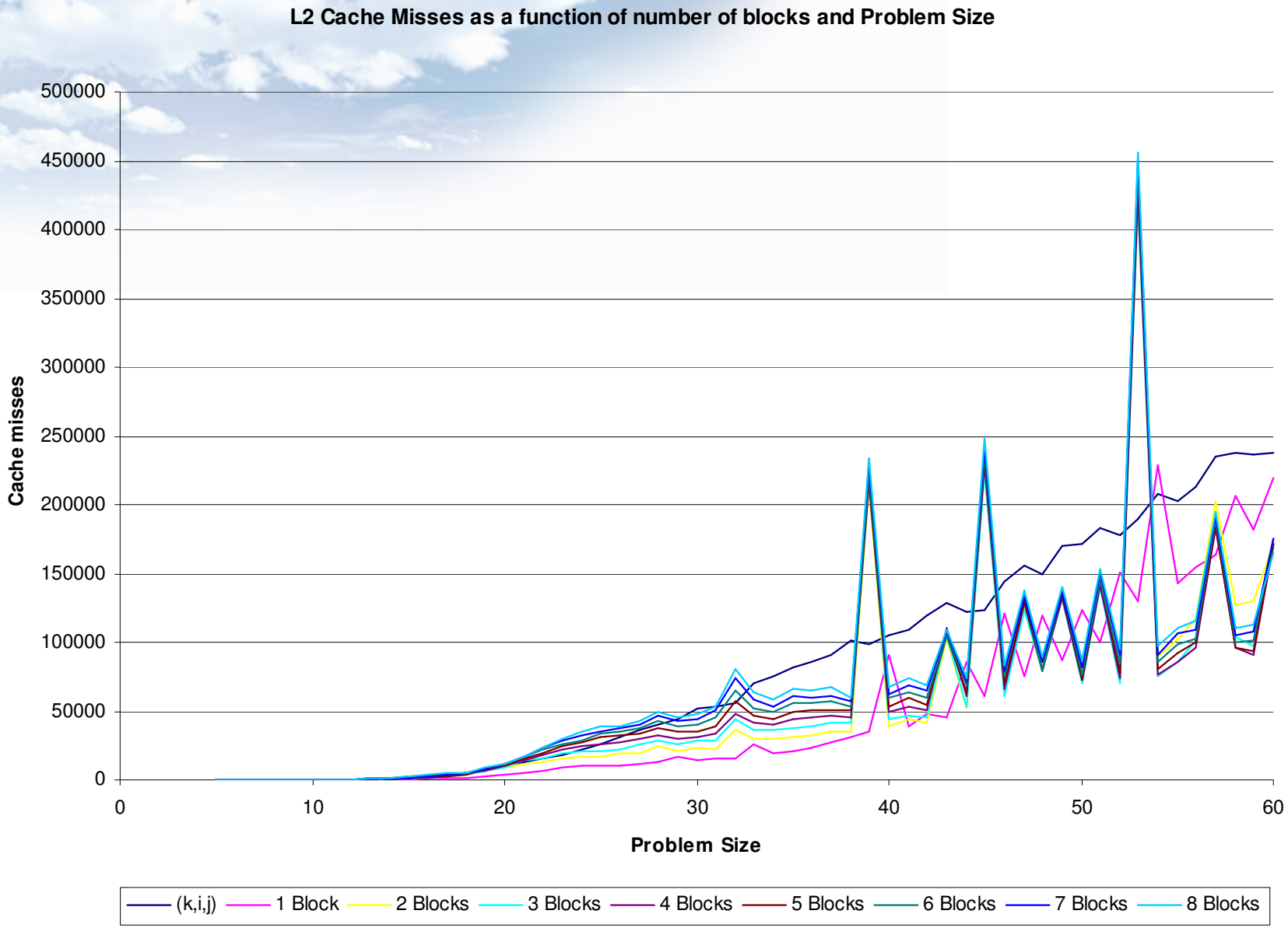
Cache Misses on Version 1



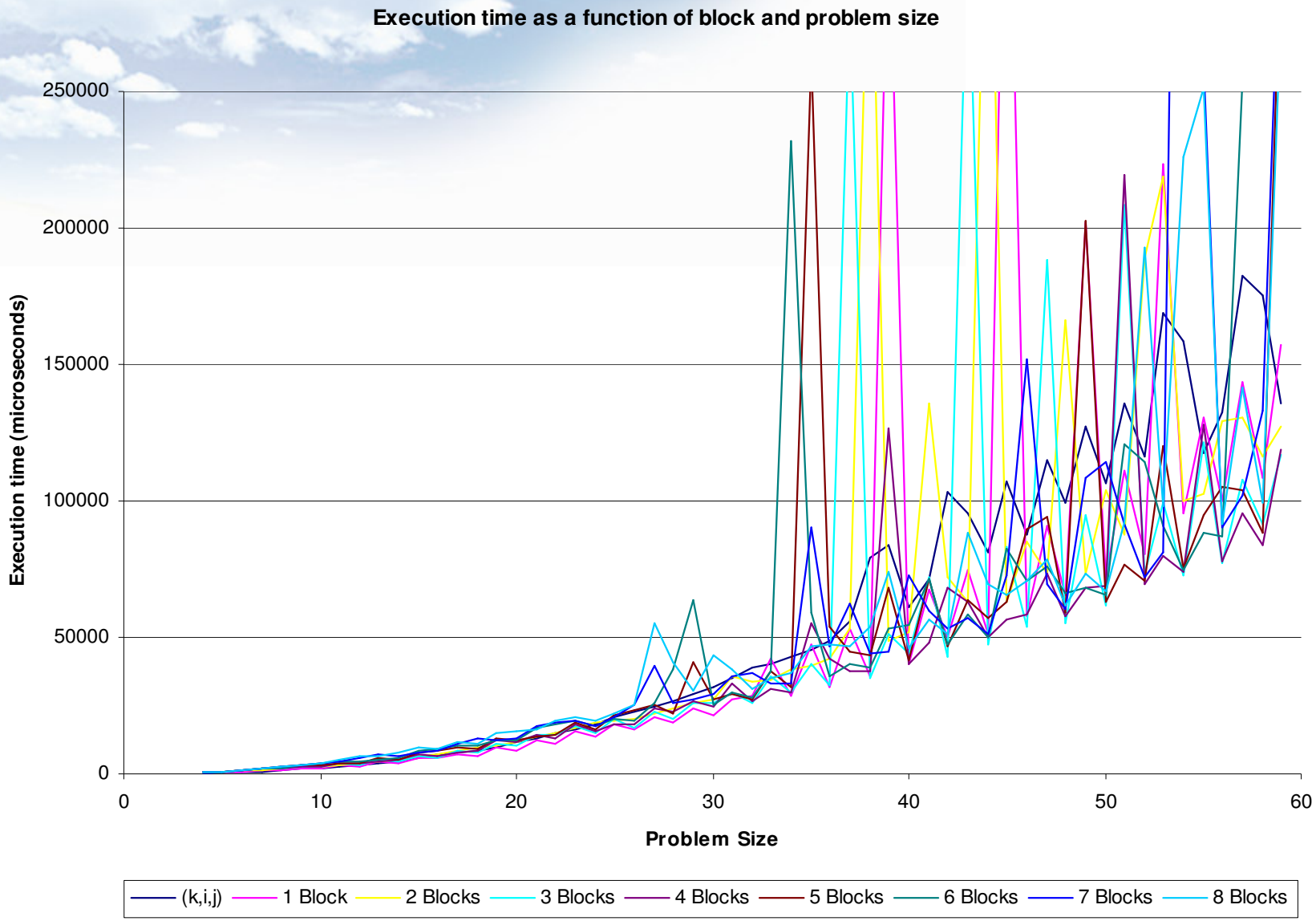
Version 2 – ijStart : ijEnd



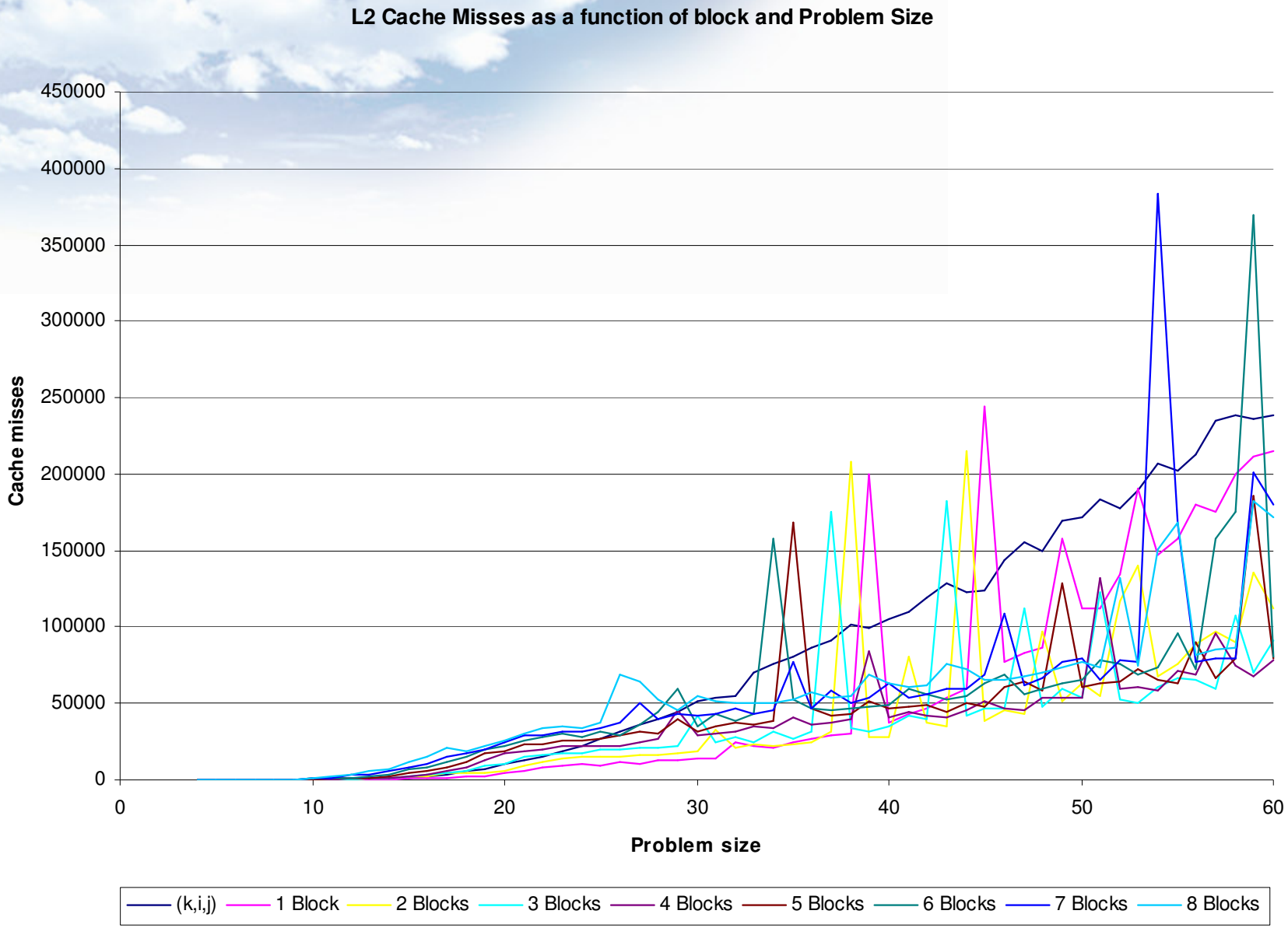
Cache Misses on Version 2



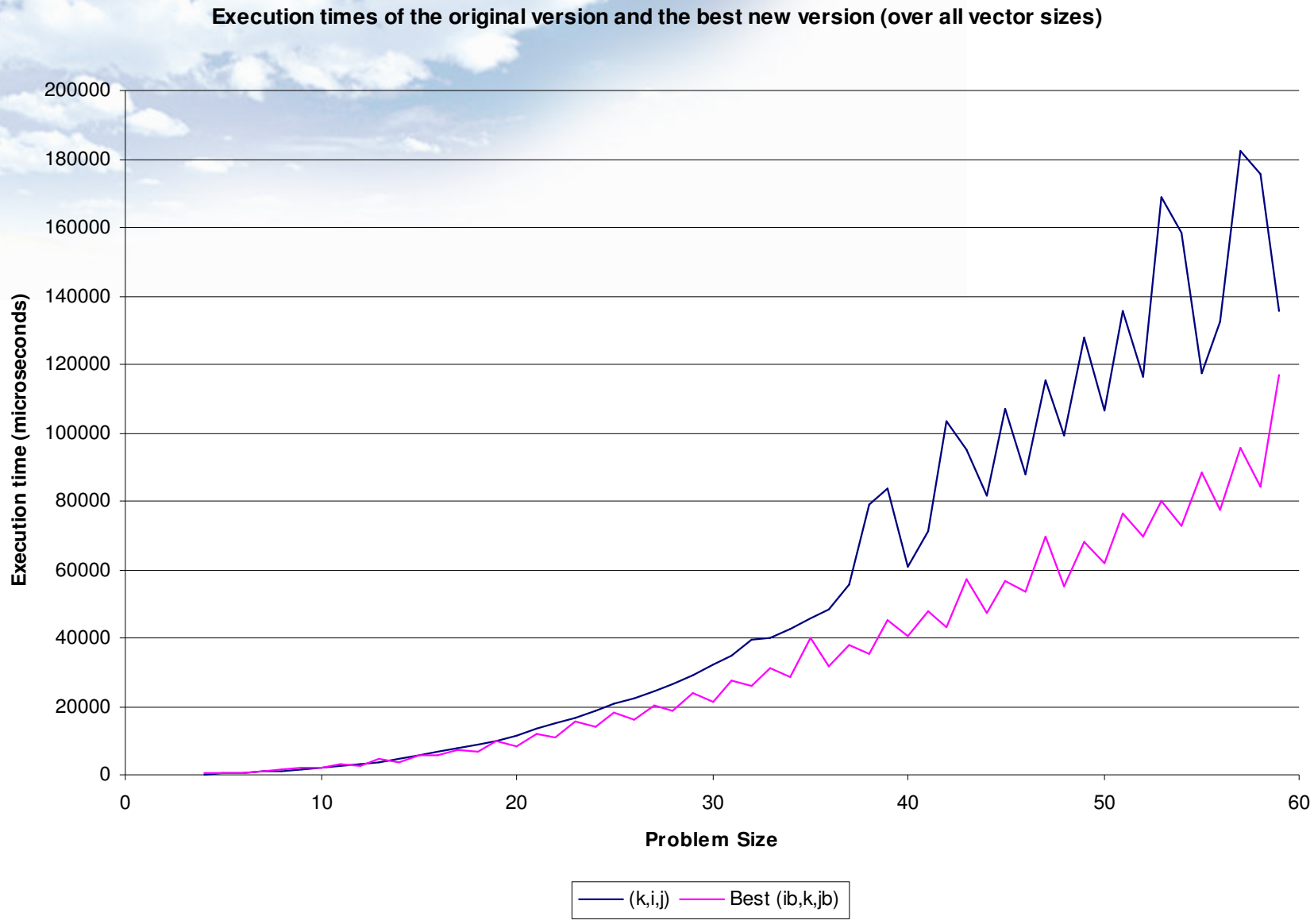
Version 3 – (ib, k, jb)



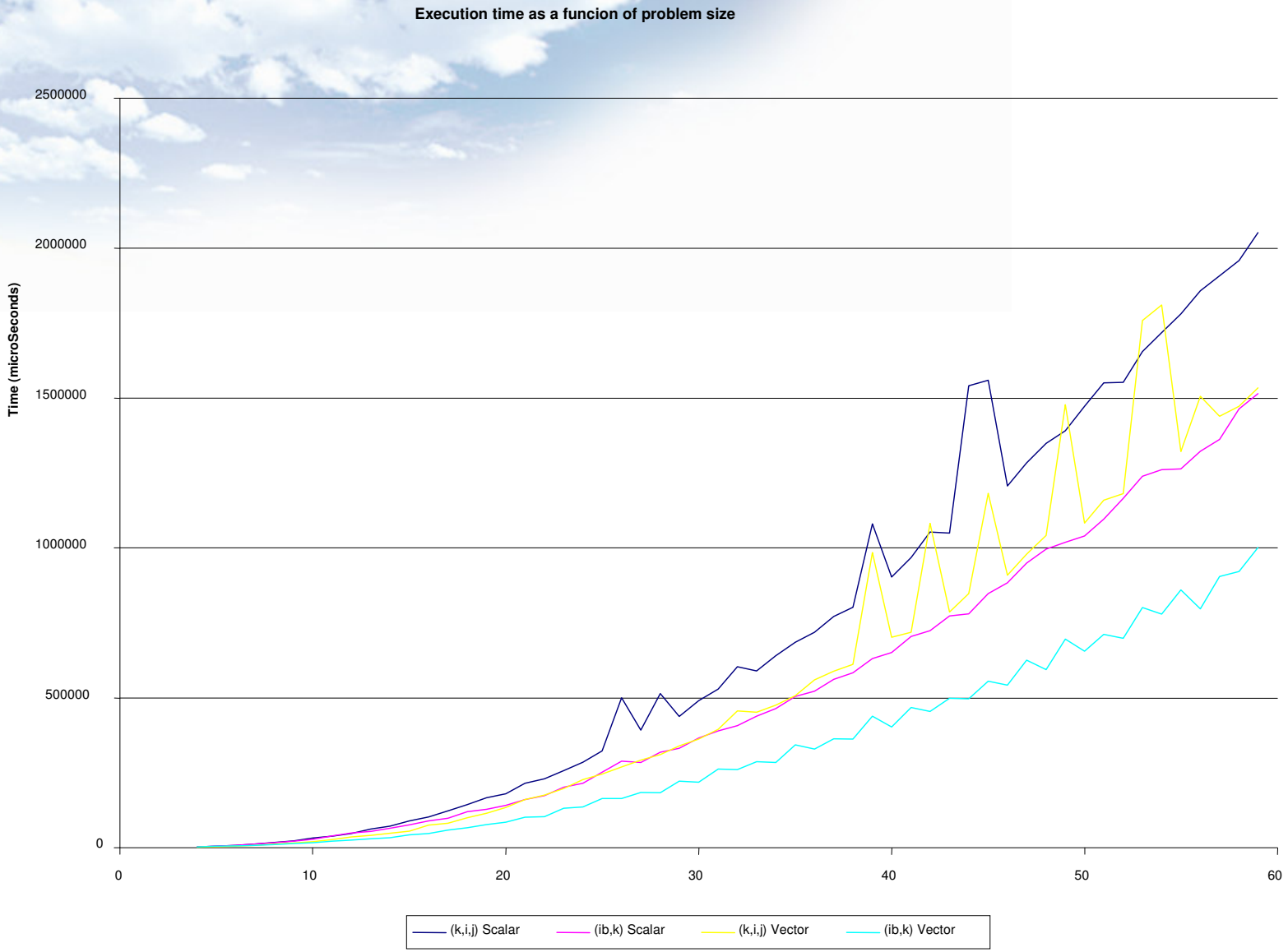
Cache Misses on Version 3



Version 3 – Best Block Size



Version 4 – Type (jb), Components (ib,k)



ij x k formulations:

- The ij formulation requires more memory
  - Potentially worst cache use, if “blind” vectorization
- But there are other dimensions besides i, j, k:
  - Frequency ranges on radiation;
  - Species on microphysics
- Repeated computations encompassing these dimensions are easier to detect on the ij formulation
  - Can be “factor out”
  - Recast the module by blocking on these dimensions

Summary

- Portability with efficiency stills an elusive goal
  - Ongoing trend for the last 30 years
- Vectorization has the potential to be a key feature to achieve this goal, but:
  - Cache reuse is central
  - Resulting code is harder to modify and understand
- Dynamic load balancing is required
- Sequential optimization is required

PART 3:
BRAMS Future Work

Diagnostics

- Current BRAMS Software Structure is inconvenient
  - Prohibitive maintenance cost
  - Lacks “state of computation” concept
  - Persistence of variables is not clear
  - Sequential optimization is badly needed
  - Weak parallel scaling
  - Prevents use of shared memory parallelism
  - Dynamic Load Balancing very hard to achieve

Proposal

- Shared Memory Parallelism should be explored
  - Due to multi-core processors (on PC clusters) and multiple processors on a SX-6 node
  - Requires extensive rewriting:
    - No hidden history carrying constructs
    - No shared scratch area
- Proposal: Two tier parallelism:
  - MPI at current level
  - OpenMP at timestep level
    - Conceptually, can be done entirely at timestep
- Proposal: Dynamic load balancing at each parallel level

Proposal

Level 1

(multiple grids, i/o, MPI Partition and load balancing)

Level 2

(timestep, OpenMP Parallelism and load balancing, MPI communication)

Level 3

(thread safe dynamics and physics modules)

Invocation

Classes of Variables

Value Carrying among time steps

(state of atmosphere + module tables)

{type at level 1 modules}

Value Carrying within a timestep

(temporaries produced / consumed within a timestep)

{automatic variables within timestep}

Within a module

(temporaries produced / consumed within a module)

{automatic variables within a module procedure}

Level 3 modules

- Thread-safe F90 Modules
  - Operates on any set of atmospheric columns
  - Blocks on non-physical dimensions
  - I/O on specific procedures, invoked outside module
  - MPI Communication on specific procedures, invoked outside module
  - Self-contained module with respect to variables
  - Input/Output procedure arguments to access external variables
  - Type encompass value carrying internal variables (ex. tables)
  - Type Creation and Destruction
  - No saved internal variables
  - Automatic arrays for internal variables
  - Fields (ij,k)

Level 2 module (timestep)

- Invokes Level 3 modules
- Operates on an MPI domain partition
  - Any set of atmospheric columns (non-rectangular)
- OpenMP parallelism
  - Creates OpenMP tiles on MPI domain partition
  - Dynamic load balancing within tiles

Level 1 procedures

- Invokes Level 2 modules
- MPI domain partition and load balancing
  - Any set of atmospheric columns (non-rectangular)
- Quite similar to today's, with extended functionalities for dynamic load balancing