

Auto-tuning Communication Operations and Libraries: Challenges and Solutions

Edgar Gabriel

Parallel Software Technologies Laboratory
Department of Computer Science
University of Houston
gabriel@cs.uh.edu

Background

- At ICL from Jan 2003 until Aug 2004
- Working on FT-MPI, Harness, Open MPI with Jack, Graham, George, Thara, Jelena, Jeffrey
- after a short interlude back in Germany now at the University of Houston since 2005

Hurricane Rita

From Wikipedia, the free encyclopedia



This article **needs attention from an expert on the subject**. See the [talk page](#) for details. [WikiProject Tropical cyclones](#) or the [Tropical cyclones Portal](#) may be able to help recruit an expert. *(December 2009)*

For other storms of the same name, see [Tropical Storm Rita](#).

Hurricane Rita was the fourth-most intense [Atlantic hurricane](#) ever recorded and the most intense [tropical cyclone](#) ever observed in the [Gulf of Mexico](#). Rita caused \$11.3 billion in damage on the [U.S. Gulf Coast](#) in September 2005.^[1] Rita was the seventeenth named storm, tenth hurricane, fifth major hurricane, and third Category 5 hurricane of the historic [2005 Atlantic hurricane season](#).

Rita made landfall on September 23 between [Sabine Pass, Texas](#), and [Johnsons Bayou, Louisiana](#), as a Category 3 hurricane on the [Saffir-Simpson Hurricane Scale](#). It continued on through parts of southeast Texas. The [storm surge](#) caused extensive damage along the Louisiana and extreme southeastern [Texas](#) coasts and destroyed some coastal communities. The storm killed seven people directly; many others died in evacuations and from indirect effects.^[2]

Contents [\[hide\]](#)

- [Meteorological history](#)

Hurricane Rita

Category 5 hurricane (SSHS)



Hurricane Ike

From Wikipedia, the free encyclopedia

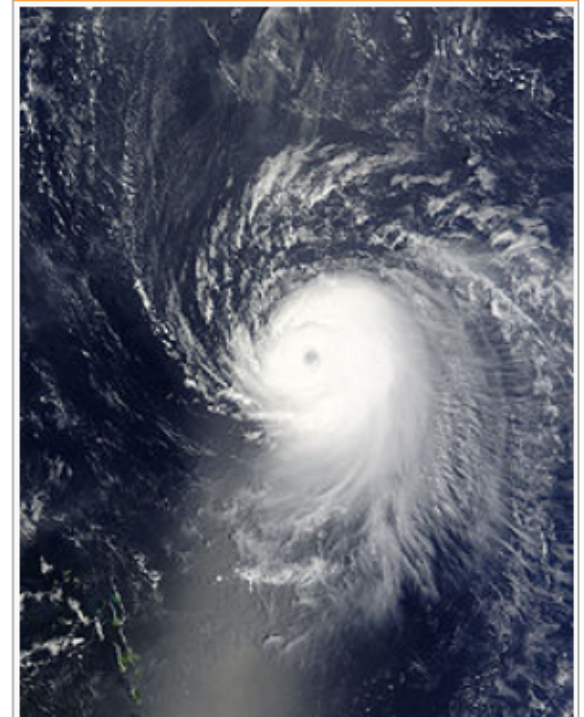
This article is about the Atlantic hurricane of 2008. For other storms of the same name, see [Tropical Storm Ike](#).

Hurricane Ike (pronounced /ˈaɪkə/) was the third costliest hurricane to ever make landfall in the United States. It was the ninth named storm, fifth [hurricane](#) and third major hurricane of the [2008 Atlantic hurricane season](#).^{[1][2]} It was a [Cape Verde-type hurricane](#), as it started as a tropical disturbance near Africa at the end of August. On September 1, 2008, it became a tropical storm west of the [Cape Verde](#) islands.^{[3][4]} By the early morning hours of September 4, Ike was a Category 4 hurricane, with [maximum sustained winds](#) of 145 mph (230 km/h) and a pressure of 935 mbar (27.61 inHg).^[5] That made it the most intense Atlantic storm of 2008. Ike passed over the [Turks and Caicos Islands](#) as Category 4, with winds 135 mph (217 km/h) on September 7. Moving west along Cuba, it made 2 landfalls as a Category 4 hurricane on September 7 and a Category 1 hurricane on September 9. Ike made its final landfall over [Galveston, Texas](#) as a strong Category 2 hurricane, with Category 5 equivalent storm surge, on September 13, 2008 at 2:10 a.m. CDT. Hurricane-force winds extended 120 miles (193 km) from the center.^[inconsistent]

Ike was blamed for at least 195 deaths. Of these, 74 were in [Haiti](#), which was already trying to recover from the impact of three storms earlier that year: [Fay](#), [Gustav](#), and [Hanna](#). In the United States, 112 people were killed, and over 300 are still missing.^{[2][6]} Due to its immense size, Ike caused devastation from the

Hurricane Ike

Category 4 hurricane (SSHS)

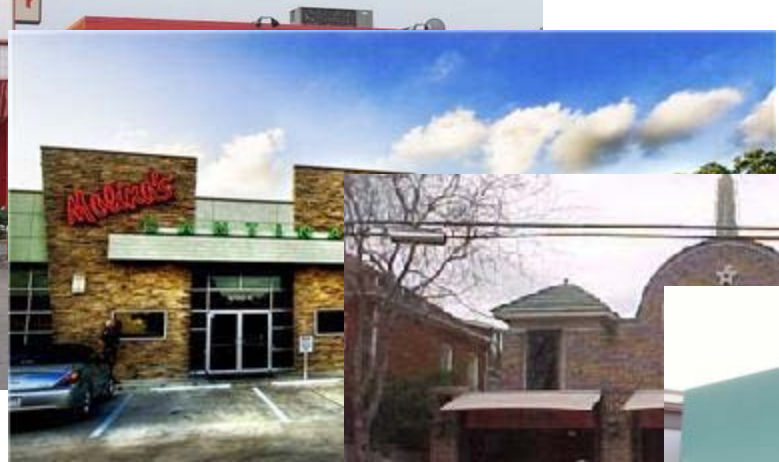




Auto-tuning \neq Auto tuning



Auto-tuning



Problems tuning parallel, collective operations

- Variety of available hardware components
 - Processor: architecture, frequency, number and organization of cores
 - Memory: speed, size, hierarchies
 - Network: performance, characteristics, topology, protocol
- Variety of software components
 - Operating system, device drivers
 - (MPI) communication library
 - Application: problem size, communication frequencies
- External influences:
 - Process placement by batch scheduler
 - Shared resources: network, file system
 - Disturbances: OS jitter, process arrival patterns

- **Problem:** How can an (average) user understand the
 - Myriad of implementation options
 - Correlations between different hardware/software components and
 - Impact of various components on the performance of the application?
- **(Honest) Answer:** no way
 - Abstract interfaces for application level communication operations required → ADCL
 - Statistical tools required to detect correlations between parameters and application performance

ADCL - Abstract Data and Communication Library

- Goals:
 - Provide abstract interfaces for often occurring (application level) collective operations
 - Provide a wide variety of implementation possibilities for each operation
 - Provide algorithms to choose the fastest available implementation at **runtime**
- Not replacing MPI, but add-on functionality
 - Uses many features of MPI

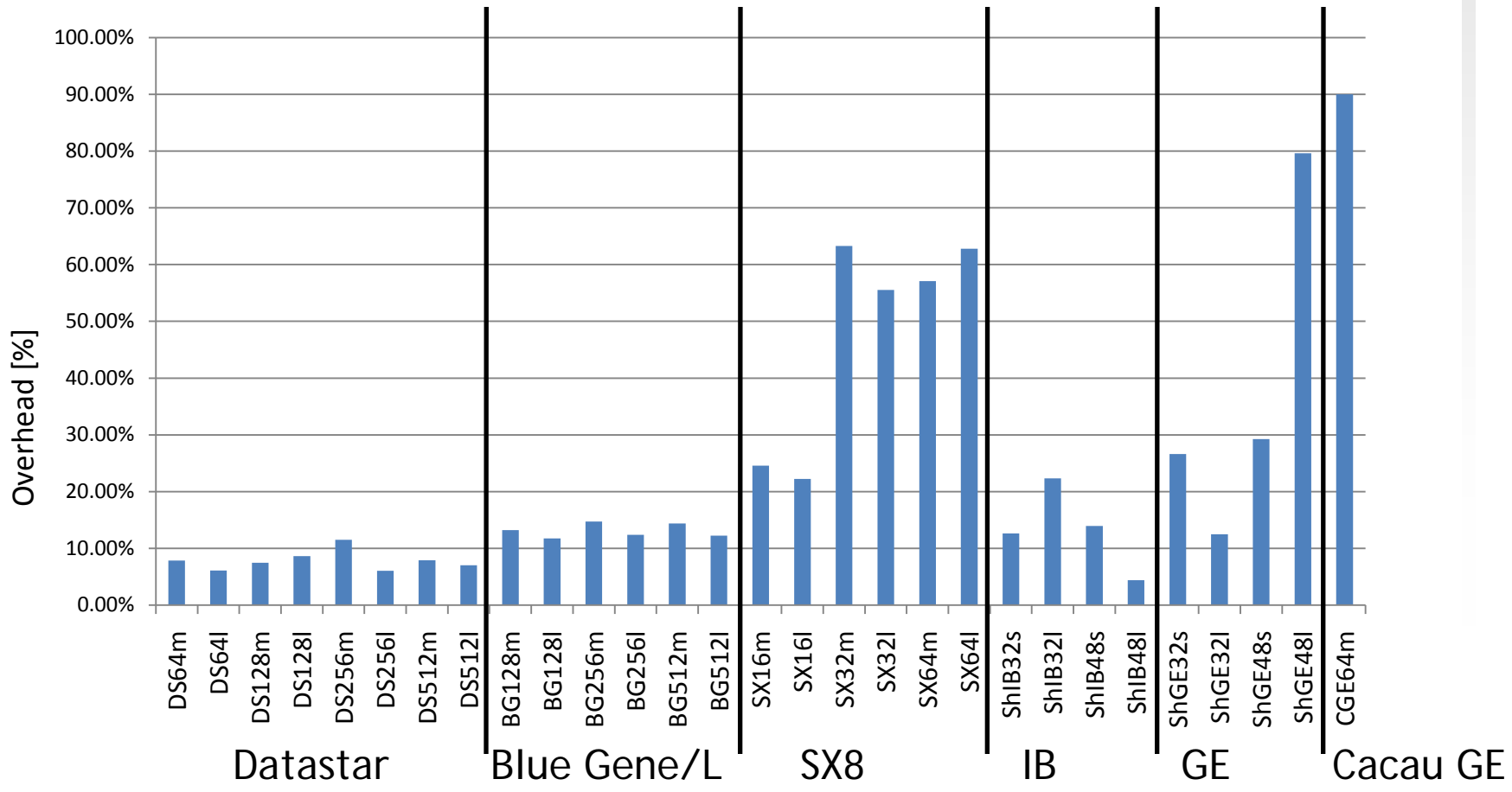
Neighborhood communication

Name	Comm. pattern	Handling of non-cont. data	Data transfer primitive
Isendrecv_aao	aao	ddt	MPI_Isend/Irecv/Waitall
Isendrecv_pair	pair	ddt	MPI_Isend/Irecv/Waitall
Sendrecv_aao	aao	ddt	MPI_Send/Irecv/Waitall
Sendrecv_pair	pair	ddt	MPI_Send/Irecv/Wait
Isendrecv_aao_pack	aao	ddt	MPI_Isend/Irecv/Waitall
Isendrecv_pair_pack	pair	pack/unpack	MPI_Isend/Irecv/Waitall
Sendrecv_aao_pack	aao	ddt	MPI_Send/Irecv/Waitall
Sendrecv_pair_pack	pair	pack/unpack	MPI_Send/Irecv/Wait
SendRecv_pair	pair	ddt	MPI_Send/Recv
Sendrecv_pair	pair	ddt	MPI_Send/Recv
SendRecv_pair_pack	pair	pack/unpack	MPI_Send/Recv
Sendrecv_pair_pack	pair	pack/unpack	MPI_Send/Recv
WinfencePut_aao	aao	ddt	MPI_Put/MPI_Win_fence
WinfenceGet_aao	aao	ddt	MPI_Get/MPI_Win_fence
PostStartPut_aao	aao	ddt	MPI_Put/MPI_Win_post/start
PostStartGet_aao	aao	ddt	MPI_Get/MPI_Win_post/start
WinfencePut_pair	pair	ddt	MPI_Put/MPI_Win_fence
WinfenceGet_pair	pair	ddt	MPI_Get/MPI_Win_fence
PostStartPut_pair	pair	ddt	MPI_Put/MPI_Win_post/start
PostStartGet_pair	pair	ddt	MPI_Get/MPI_Win_post/start

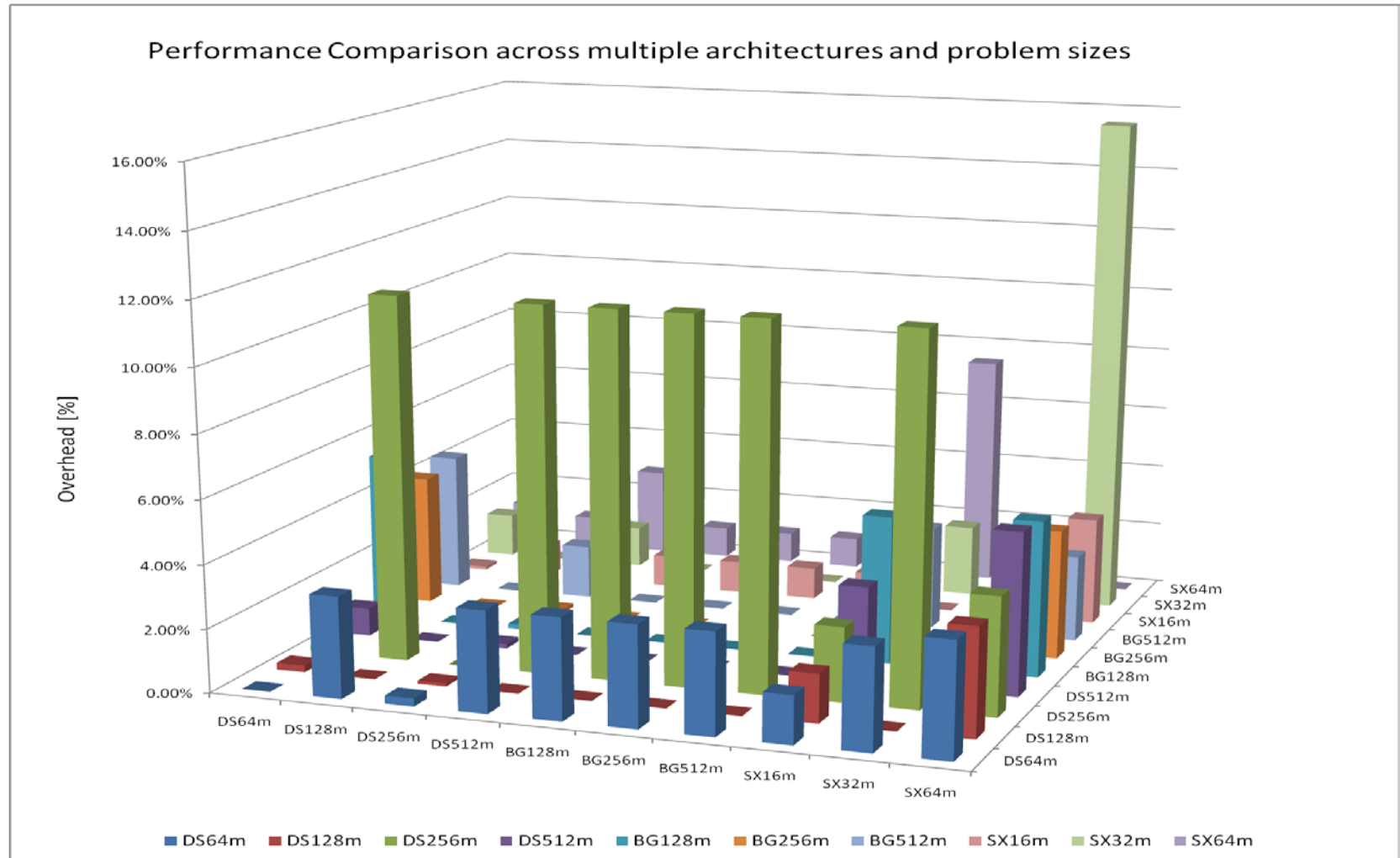
Architecture	# of proc	Pb Size/Proc	Best implementation
DataStar p655	64	64x32x32	Sendrecv_aao
		64x64x32	Sendrecv_aao
	128	64x32x32	Sendrecv_pair_pack
		64x64x32	Isendrecv_aao
	256	64x32x32	Isendrecv_aao
		64x64x32	Isendrecv_aao
	512	64x32x32	Sendrecv_pair_pack
		64x64x32	Isendrecv_aao
IBM Blue Gene/L	128	64x32x32	Sendrecv_pair_pack
		64x64x32	Isendrecv_aao_pack
	256	64x32x32	Sendrecv_pair_pack
		64x64x32	Isendrecv_aao_pack
	512	64x32x32	Sendrecv_pair_pack
		64x64x32	Isendrecv_aao_pack
NEC SX8	16	64x32x32	Sendrecv_pair
		64x64x32	Sendrecv_pair
	32	64x32x32	Sendrecv_pair_pack
		64x64x32	Sendrecv_pair_pack
	64	64x32x32	Sendrecv_pair_pack
		64x64x32	Sendrecv_pair_pack
SharkIB	32	32x32x32	Isendrecv_aao
		64x64x32	Sendrecv_pair_pack
	48	32x32x32	Sendrecv_aao
		64x64x32	Isendrecv_aao
SharkGE	32	32x32x32	Isendrecv_aao
		64x64x32	Isendrecv_aao_pack
	48	32x32x32	Isendrecv_aao
CacauGE	64	64x64x32	Sendrecv_pair
		64x32x32	SendRecv_pair_pack

Penalty of using a hard-coded implementation

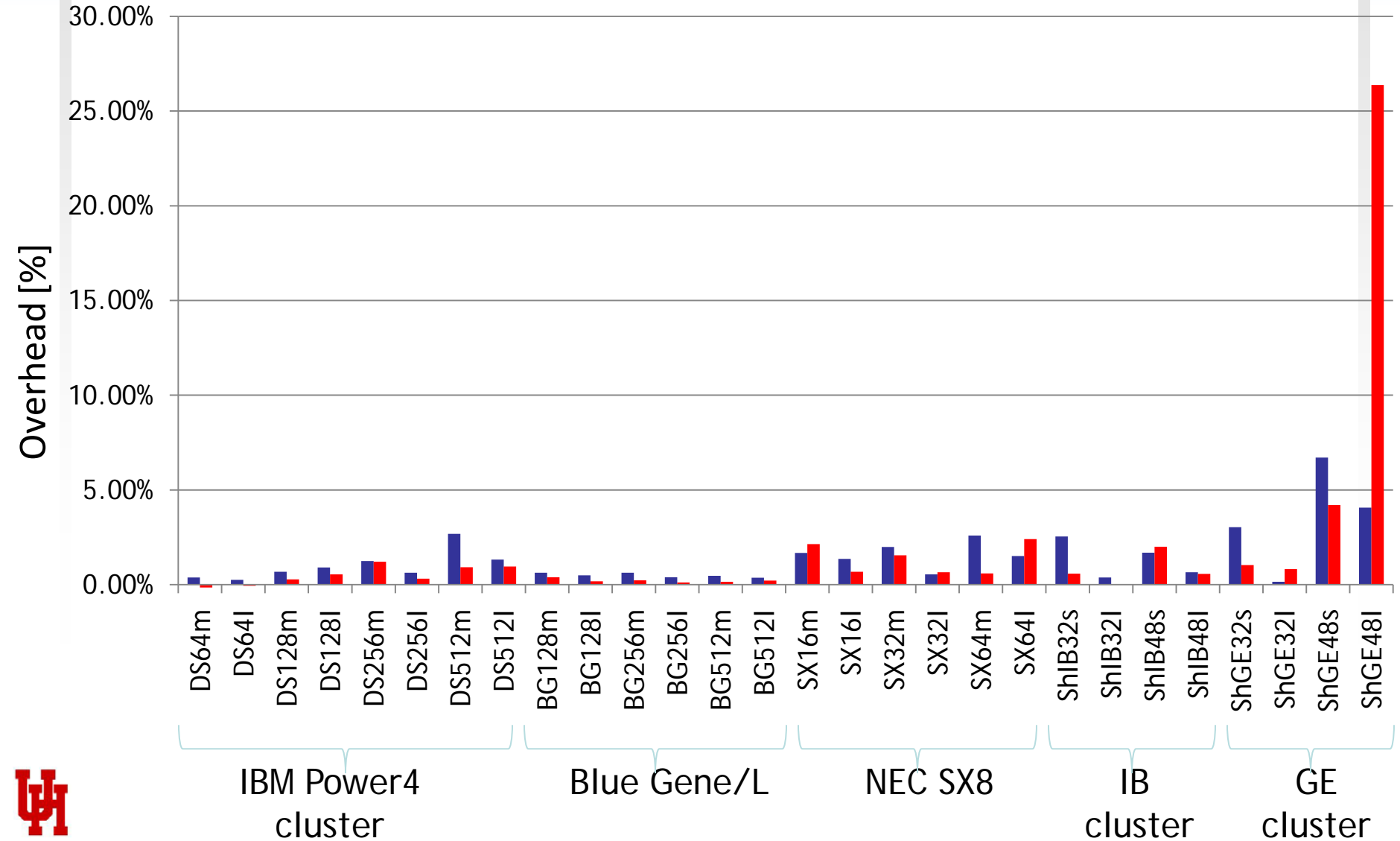
Overhead of worst vs. best performing implementation for each platform and problem size



Penalty of using a pre-tuned solution of a platform on another platform (II)



■ ADCL brute force search ■ ADCL attribute based search



Auto-tuning revisited



Historic learning (I)

Motivation

- A large class of HPC applications show a dynamic behavior with respect to problem sizes
 - Example: mesh refinement in CFD simulations
- Performance overhead introduced by ADCL by running extremely slow implementations during the selection logic
 - Example: 72% of performance penalty in a cluster with hierarchical network

Historic learning (II)

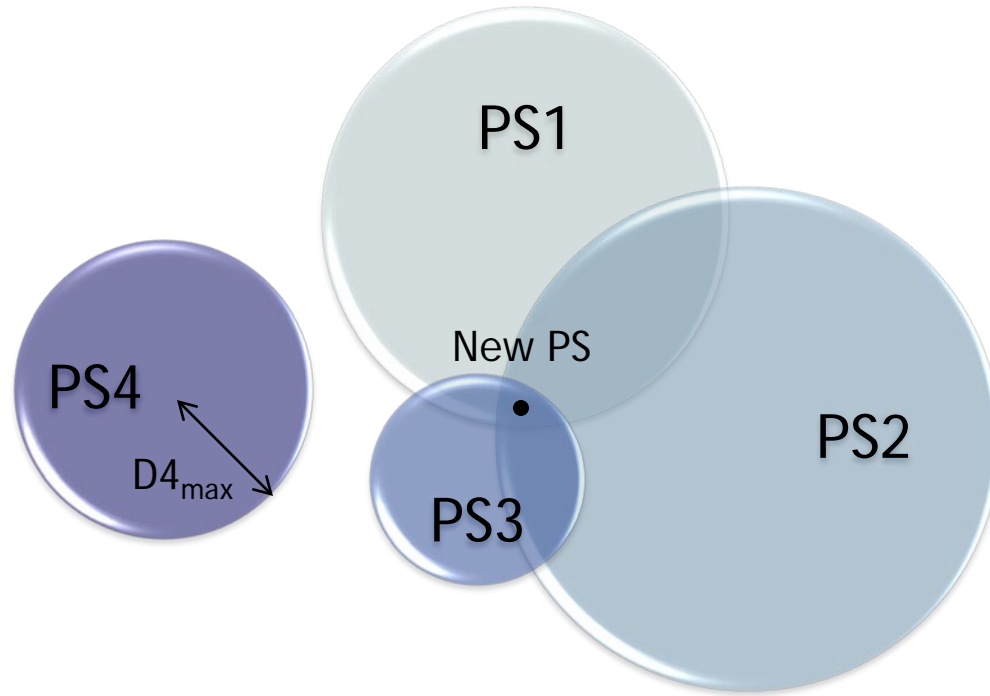
General concept

- Once the best implementation for a particular problem is selected using a runtime selection algorithm, the problem characteristics and its solution are stored in a history file
- Reuse the knowledge learned from previous problems to predict a best or near to the best implementation for identical/similar problems

Historic data analysis (II)

- Distance measure defined by the function-set
 - e.g. Euclidian Distance between the message length to be transferred to each neighbor for the n-Dimensional neighborhood communication function-set
- For each entry, compute D_{\max} : the distance within which all other entries or most of them are related
 - Defines the range to which the winner implementation can be applied without dramatic performance degradation

Historic Learning



Prediction Algorithms

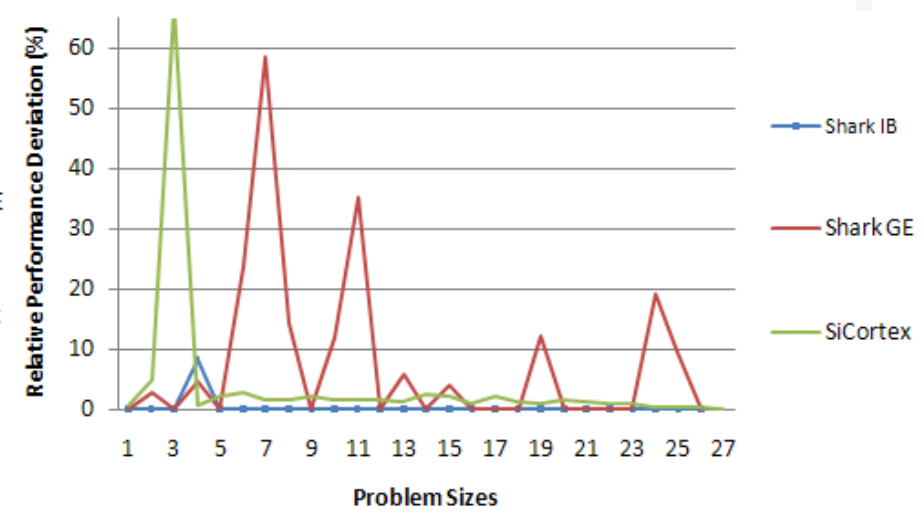
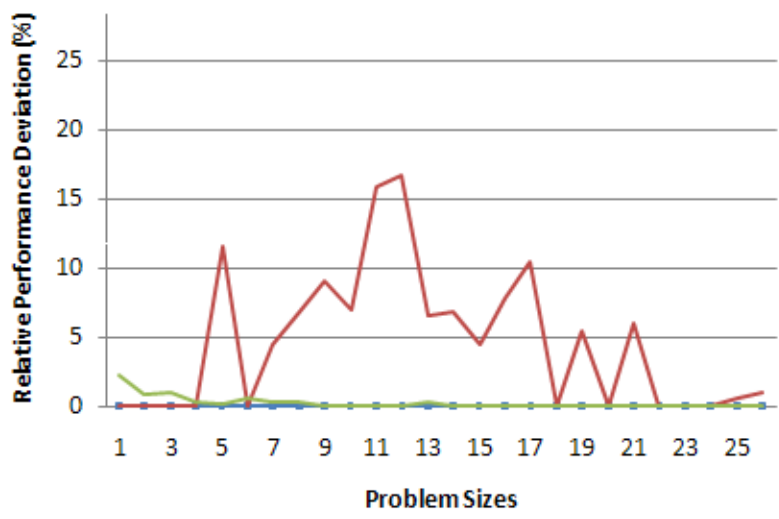
- **Algorithm 1: PredictFromClosest** predicts the code version by taking the winner version of the closest problem size in the history file if it is within its maximum distance
- **Algorithm 2: PredictFromSimilar** suggests to use the winner of a weighted majority vote from the similar problem sizes for which it is within their defined maximum distance

Experimental results (I)

- Adaptive applications simulated by 26 different problem sizes to be solved successively
- 3D neighborhood communication pattern
- Tests executed on a cluster using either infiniband or Gigabit Ethernet interconnect and a SiCortex system
- Historic file containing 42-50 entries
 - Not necessarily very accurate
 - A smoothing operation, with different window sizes used to remove the outlier entries
- Tests with different acceptable performance windows

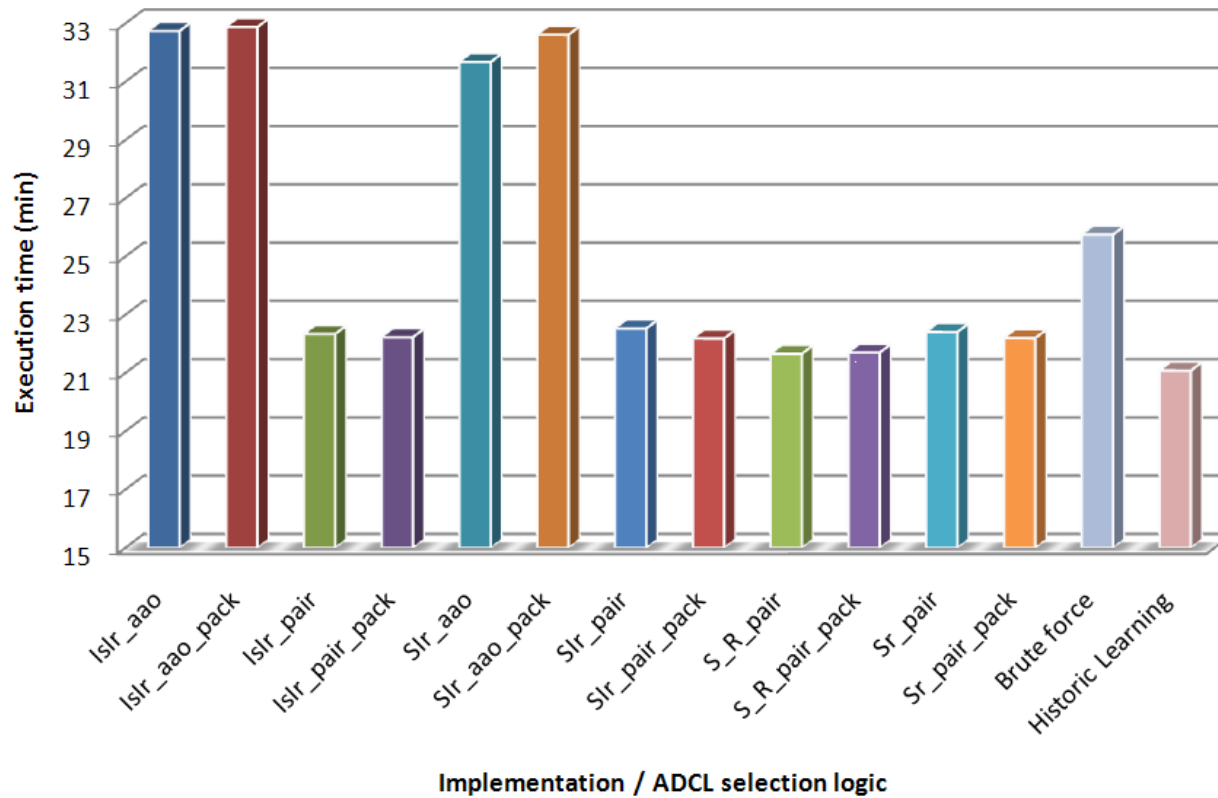
Experimental results (II)

- Prediction accuracy using two CFD kernels refining the mesh 26 times on three platforms
- Relative Performance Deviation to the optimal implementation for that problem size
 - 0% = optimal implementation



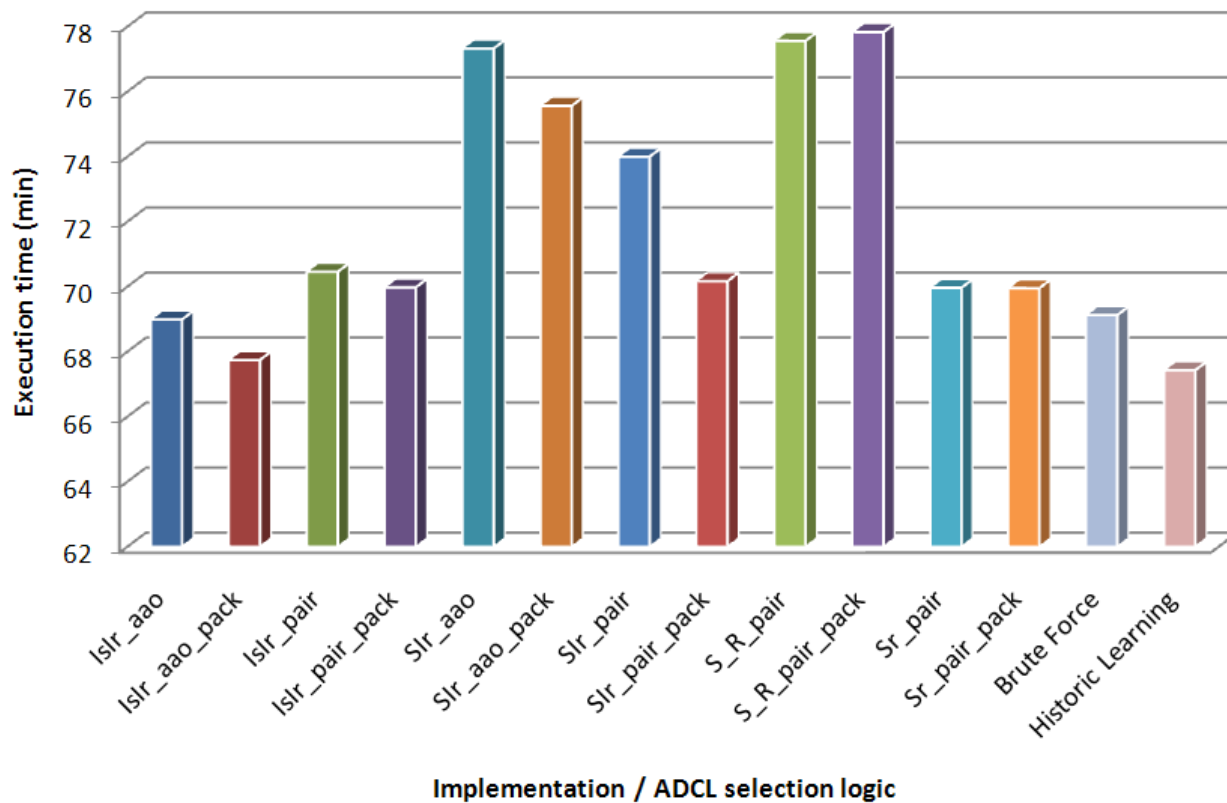
Experimental results (III)

Performance Benefit (Solversys GE)



Experimental results (IV)

Performance Benefit (Parheat SiCortex)



2^k factorial design

Objective: perform the minimum number of evaluations and still obtaining the best or close to the best parameter combination

- Evaluate the 2^k code versions corresponding to the outermost values of parameter combinations
- Using a non-linear regression model, an indication on the proportion of the total variation in the performance that is explained by each factor/parameter is obtained

$$p = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

- Reduce the number of factors (k) and to choose those that have significant impact on the performance

Tuning Open MPI parameters

4 parameters of the openib module:

- using NetPipe as benchmark
- 4 byte latency

	Brute force search	Attribute based search	2k Factorial Design search
Time	5h 20min 15sec	1min 52sec	32sec
#solutions found	2136	24	5
#combinations evaluated	15986	109	24

Tuning Open MPI parameters

parameters of the openib module and of the hierarch collective module

- using SkaMPI as benchmark
- broadcast operation
- 1 MByte message length
- 16 processes

# parameters	#combinations	Brute force search	Attribute based search	2k Factorial design search
3 (2+1)	84	4min 50sec	2min 5sec	1min 49sec
4 (2+2)	792	42min 43sec	3min 55sec	3min 8sec
5 (2+3)	5040	4h 34min 38sec	6min 42sec	4min 10sec

Remaining challenges

- More than I can list here, but
 - overlapping communication and computation
 - simultaneous optimization of multiple operations