Application Performance Analysis with SGI MPInside

Daniel Thomas, Jean-Pierre Panziera, John Baron
SGI Performance Engineering Team

Agenda

- MPI and MPInside overview
- Profiling capabilities
  - Basic features
  - Collective wait time evaluation
  - Send late time
- Modeling capabilities
- Case study
- Availability
- Conclusion

How important is MPI for SGI

- **The Message Passing Interface (MPI) standard**

  MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users.

- SGI is a major provider in the High Performance Processing (HPC) world with 20 machines in the top500 faster computer in the world

Rank	Site	Manufacturer	Computer	Country	Year	Cores	RMax	RPeak
4	NASA/Ames Research Center/NAS	SGI	SGI Altix ICE 8200EX, Xeon Q(	United States	2008	51200	487005	608829
17	New Mexico Computing Applications Center (N	SGI	SGI Altix ICE 8200, Xeon quad	United States	2007	14336	133200	172032
20	Grand Equipement National de Calcul Intensif -	SGI	SGI Altix ICE 8200EX, Xeon qu	France	2008	12288	128400	146736
29	Total Exploration Production	SGI	SGI Altix ICE 8200EX, Xeon qu	France	2008	10240	106100	122880
43	HLRN at Universitaet Hannover / RRZN	SGI	SGI Altix ICE 8200EX, Xeon X5	Germany	2009	7680	82570	90009,6
44	HLRN at ZIB/Konrad Zuse-Zentrum fuer Informa	SGI	SGI Altix ICE 8200EX, Xeon X5	Germany	2009	7680	82570	90009,6
55	US Army Research Laboratory (ARL)	SGI	SGI Altix ICE 8200 Enhanced L	United States	2009	6656	70000	74547,2
58	NASA/Ames Research Center/NAS	SGI	SGI Altix 1.5/1.6/1.66 GHz, Vo	United States	2008	13824	66567	82944
66	Leibniz Rechenzentrum	SGI	Altix 4700 1.6 GHz	Germany	2007	9728	56520	62259,2
75	Wright-Patterson Air Force Base/DoD ASC	SGI	Altix 4700 1.6 GHz	United States	2007	9216	51441	58982
93	National Institute for Materials Science	SGI	SGI Altix ICE 8200EX, Xeon X5	Japan	2009	4096	42690	45875,2
199	HLRN at Universitaet Hanno		SGI Altix ICE 8200EX, Xeon qu	Germany	2008	2560	26690	30720
200	HLRN at ZIB/Konrad Zuse-Z		SGI Altix ICE 8200EX, Xeon qu	Germany	2008	2560	26690	30720
222	Irish Centre for High-End Co		SGI Altix ICE 8200EX, Xeon qu	Ireland	2008	2560	25110	28672
292	RQCHP/Compute Canada		Altix XE320 Cluster, Xeon 2.8G	Canada	2009	2464	21600	27596,8
301	IFREMER		SGI Altix ICE 8200EX, Xeon X5	France	2009	2048	21345	22937,6
308	Wright-Patterson Air Force E		SGI Altix ICE 8200 Enhanced L	United States	2009	2048	21160	22937,6
356	SGI		SGI Altix ICE 8200, 3.0 GHz	United States	2007	2048	19360	24576
470	Idaho National Laboratory	SGI	SGI Altix ICE 8200, 2.66 GHz	United States	2007	2048	17780	21791
490	University of Minnesota	SGI	SGI Altix XE 1300 Cluster Solu	United States	2007	2048	17310	21791

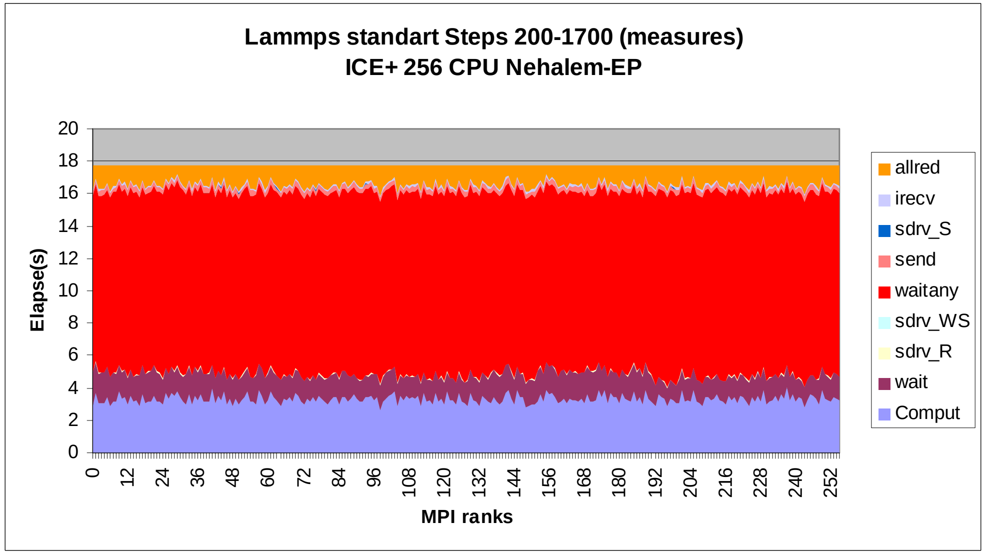
A large majority of the flops executed with these machines are executed by MPI applications

A general performance tool tradeoff

- What do we want to know with the tool
  - Data about elements that are important for the MPI library developer may be of little interest for the user that have no way to interact with such elements
- How to use the tool
  - Use the binary the way it is or change it (recompile, relink, insert calls in the source)
  - How to fire up the tool
- What to gather
  - Cumulative measurement stats
  - Modeled results
  - Traces
- What to report
  - Raw data
    - Need for powerful post-processing ?
  - Pre-interpreted data
- Amount of resources to make it

MPInside Purpose

- To gain a better understanding of the interaction between application and MPI library/interconnect network by diving “inside” the internals of each



- For the application developers to understand the consequence of their choices for exchanging data
- For our performance engineering group that needs to commit on application performance with future hardware

MPInside Design goals

- To require no re-compilation or re-linking.
- To use a simple command line interface.

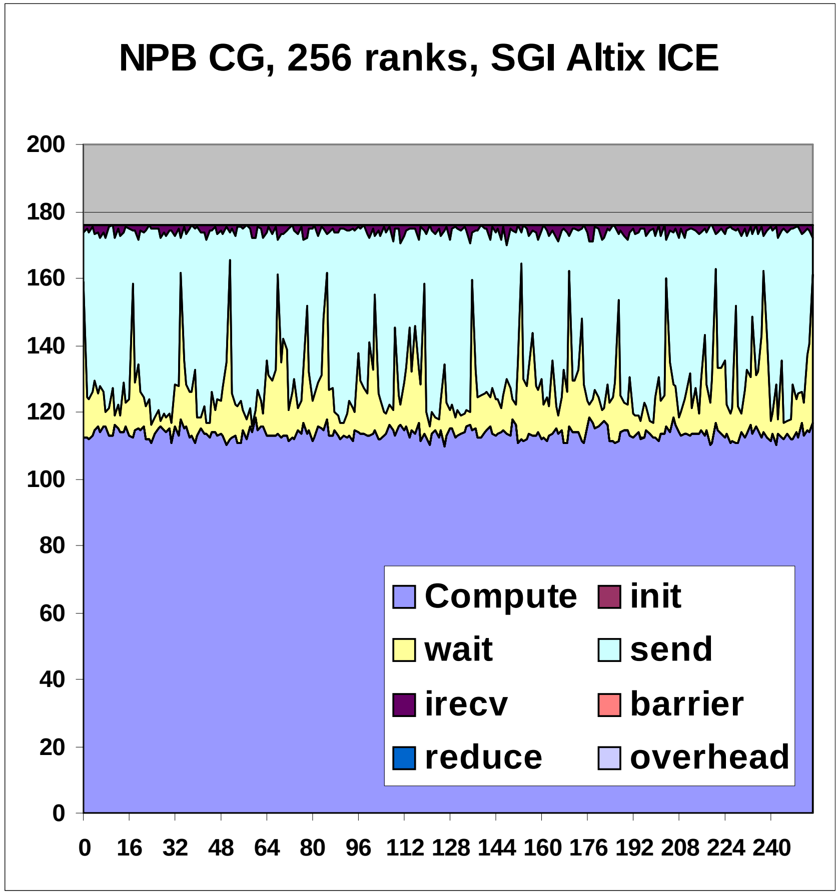
- To be useable with thousands of ranks without overhead.
- To work without traces and without post-processing.
  - This is a strong constraint that needs to use innovative and courageous solutions
- To handle various communication models, in particular the perfect interconnect (zero latency, infinite bandwidth).

- To produce simple text, easy to parse, raw output to be processed with common scripting tools(awk,..) and a spreadsheet

- To be portable to any MPI library for its basic features.
- To support the full MPI 1.2 specifications and the MPI-2 one-sided communications.

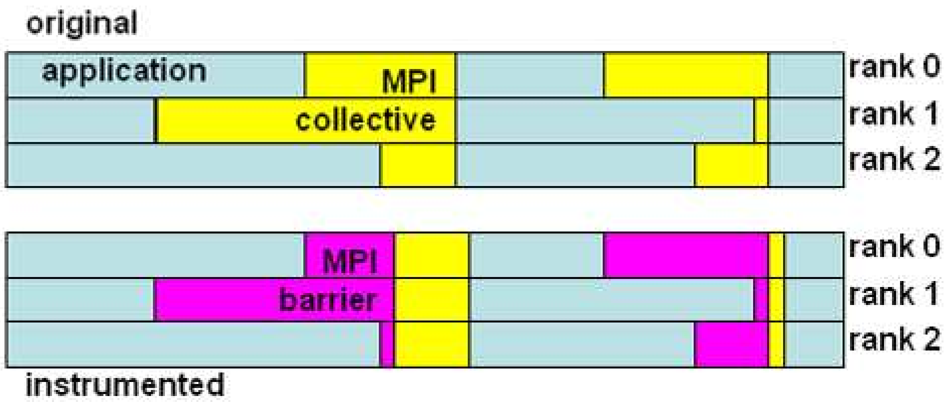
MPInside Basic Statistics

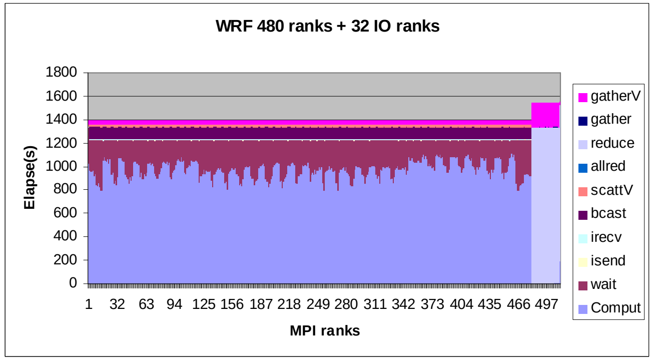
>>>>	Elapse	times	in	(s)	<<<<		
CPU	Comput	init	wait	send	irecv	bcast	overheac
0	7.7442	0.0486	2.1606	0.0000	0.0034	0.0146	0.0033
1	9.6735	0.0471	0.0000	0.2357	0.0000	0.0150	0.0020
2	7.7667	0.0458	0.2484	0.0000	0.0042	1.9045	0.0023
3	7.7251	0.0450	0.0000	0.2633	0.0000	1.9361	0.0016
>>>>	Kbytes	with	send	attribute	<<<<		
CPU	Comput	init	wait	send	irecv	bcast	overheac
0	------	0	0	0	0	0	0
1	------	0	0	400000	0	4000	0
2	------	0	0	0	0	0	0
3	------	0	0	400000	0	0	0
>>>>	Number	of	request:	with	Send	attribute<<<<	
CPU	Comput	init	wait	send	irecv	bcast	overheac
0	------	1	0	0	0	0	0
1	------	1	0	1000	0	1000	0
2	------	1	0	0	0	0	0
3	------	1	0	1000	0	0	0
>>>>	Kbytes	with	Recv	attribute	<<<<		
CPU	Comput	init	wait	send	irecv	bcast	overheac
0	------	0	0	0	400000	4000	0
1	------	0	0	0	0	0	0
2	------	0	0	0	400000	4000	0
3	------	0	0	0	0	4000	0
>>>>	Number	of	request:	with	Recv	attribute<<<<	
CPU	Comput	init	wait	send	irecv	bcast	overheac
0	------	0	1000	0	1000	1000	0
1	------	0	0	0	0	0	0
2	------	0	1000	0	1000	1000	0
3	------	0	0	0	0	1000	0



mpirun –np NNN MPInside <cmd>

Collective Wait Time



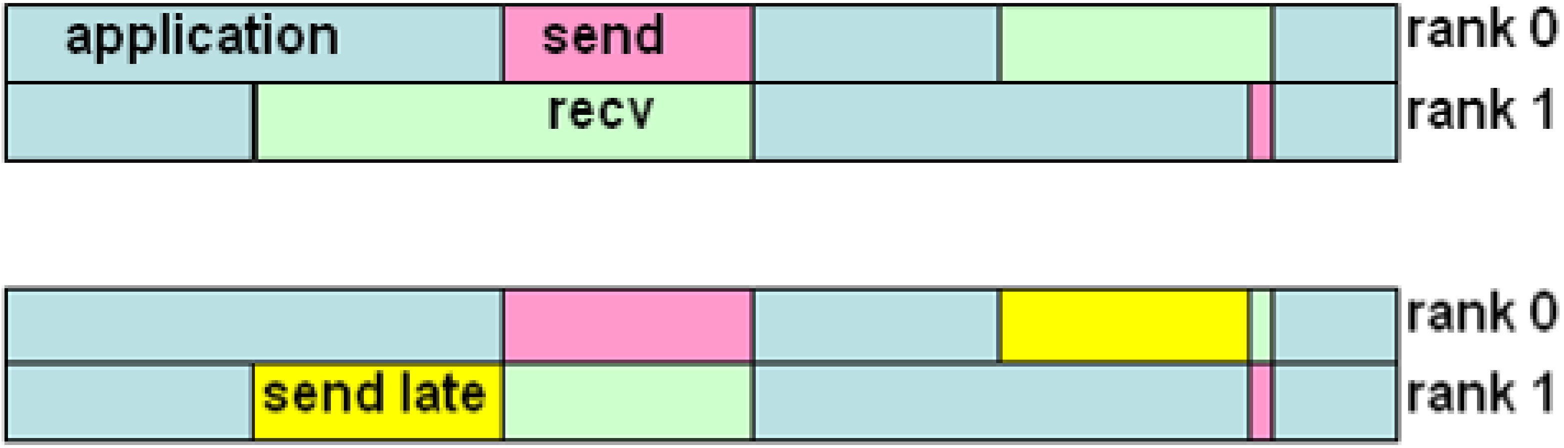


setenv MPINSIDE_EVAL_COLLECTIVE_WAIT

A simple MPI_Barrier is inserted before the collective function assuming:

<Time collective> = <time to synchronize> + <time collective with fully synchronized arrivals>

Late Senders



setenv MPINSIDE_EVAL_SLT

Calculates per-rank times when sends were late for blocking MPI functions(MPI_Recv, MPI_Wait,..)

Late Senders Time: How to capture it ?

- Have a synchronized clock (SGI Altix, UV, future ICE):
  - A mechanism to send/recv a supplemental piece of information (at least the send posting time) with the user buffer needs to be implemented
- Clocks are not synchronized and deviates (the casual situation on clusters)
  - Use a stuttering method:

    Send/recv fisrt a zero message size then the data

    Time recv = Time Waiting send is posted + time transfer

    Approximated as

    Time to get zero message size + time to recv when send is surely posted

    Only applicable with Bandwidth sensitive applications
- More on next slide

Stuttering method: Just look at my local clock

Rank 0	**Rank 1**
`1:MPI_Irecv(from 1,app tag )`	`MPI_Irecv(from 0,app tag)`
`2:MPI_send(to 1)`	`MPI_send(to 0)`
`3:MPI_Wait (recv)`	`MPI_Wait(recv)`

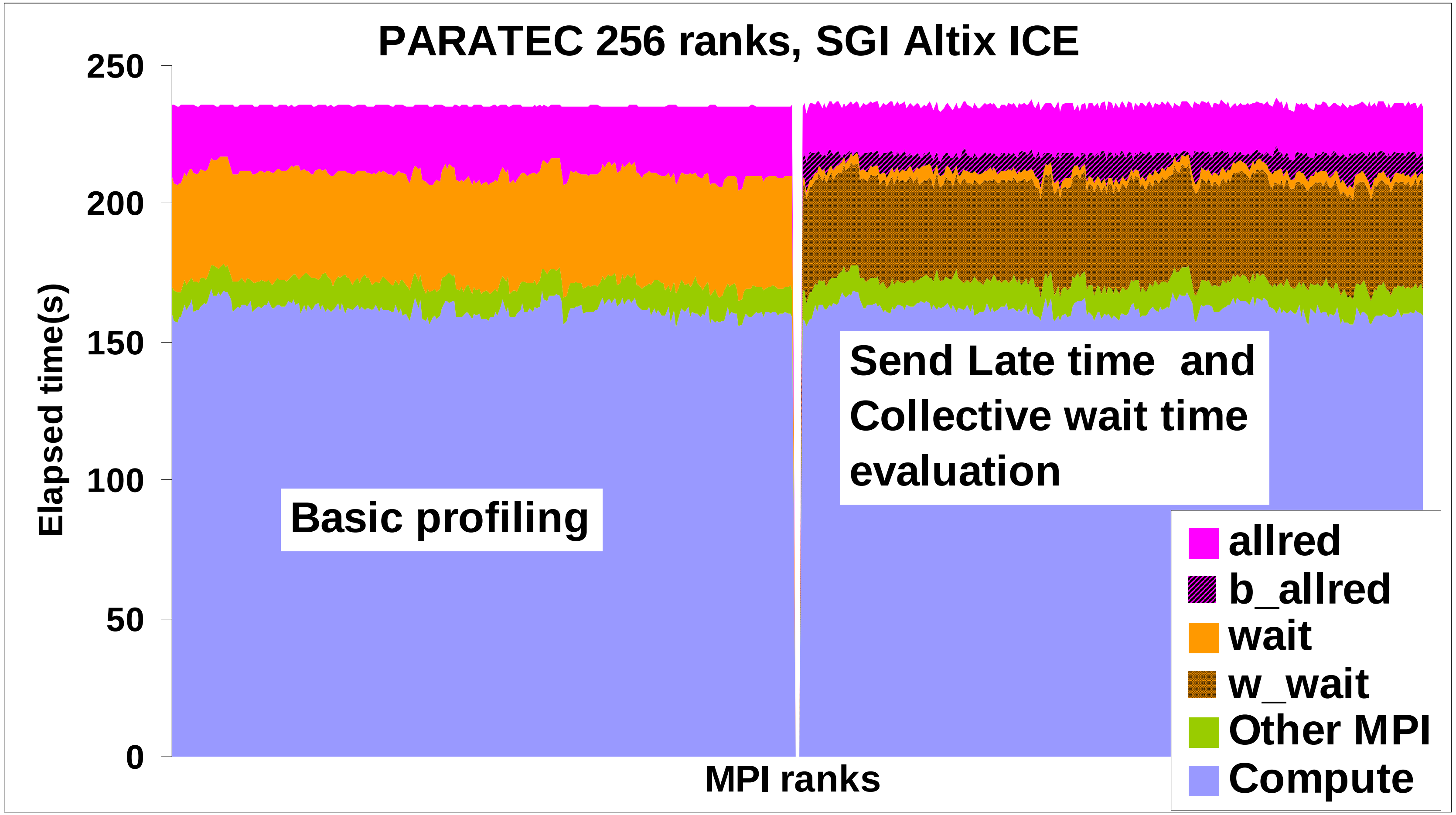
-The 3a “zero message” Time is the “Send Late Time”

Stuterring method:

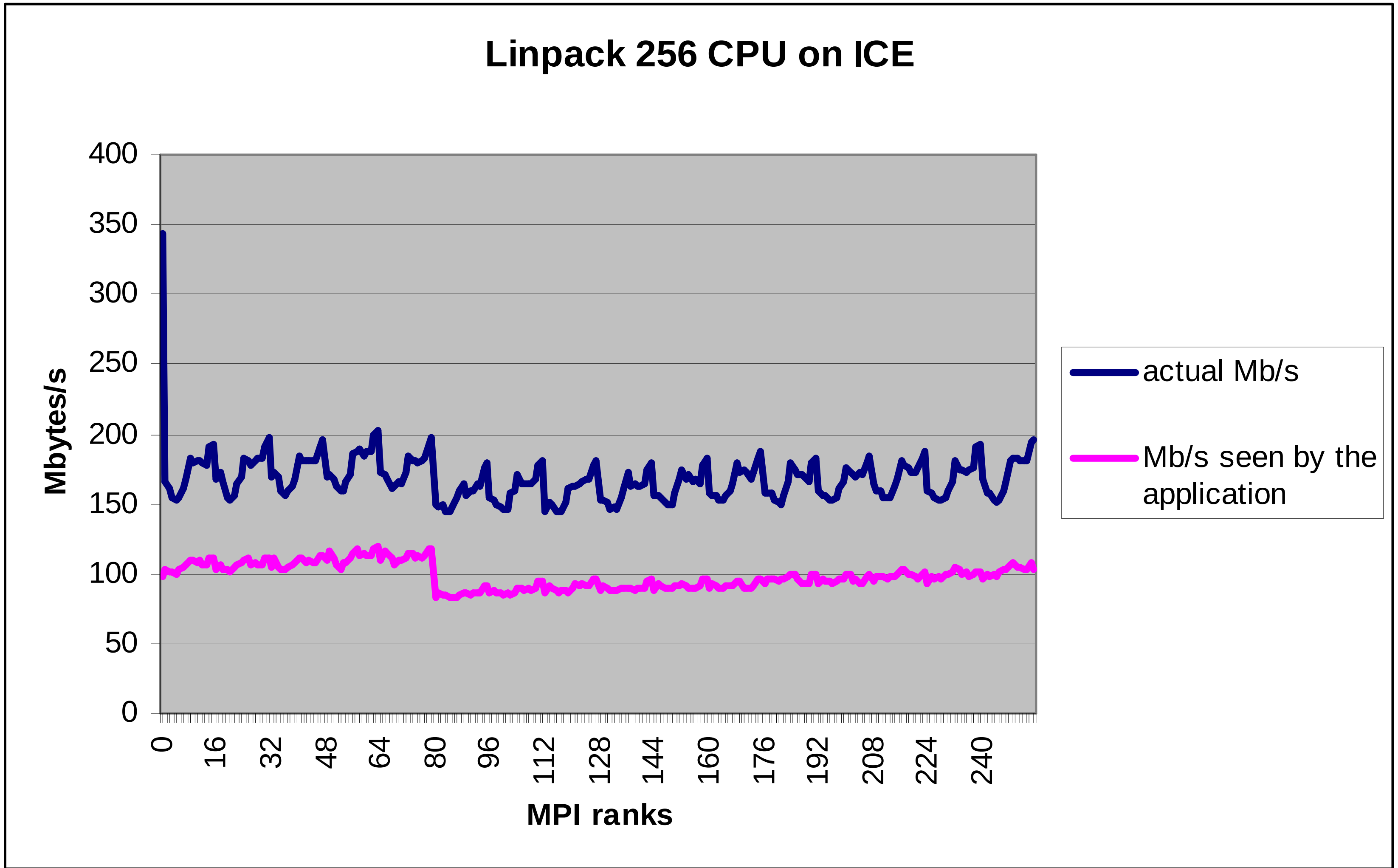
`1a:MPI_Irecv( from 1, app tag)`	`MPI_Irecv(app tag from 0)`
`2a:MPI_Isend(to 1,app tag)`	`MPI_Isend(to 0,app tag)`
`2b:MPI_Isend(to 1,data,token tag)`	`MPI_Isend(to 0,data,token tag)`
`2c:MPI_Wait (request 2a)`	`MPI_Wait(request 2a)`
`2d:MPI_Wait (request 2b)`	`MPI-Wait(request 2b)`
`3a:MPI-Wait(request 1a)`	`MPI_Wait(request 1a)`
`3b:MPI_Recv(from 1, token tag)`	`MPI_Recv(from 0,token tag)`

-The 3b time is the time of a transfer with ready send

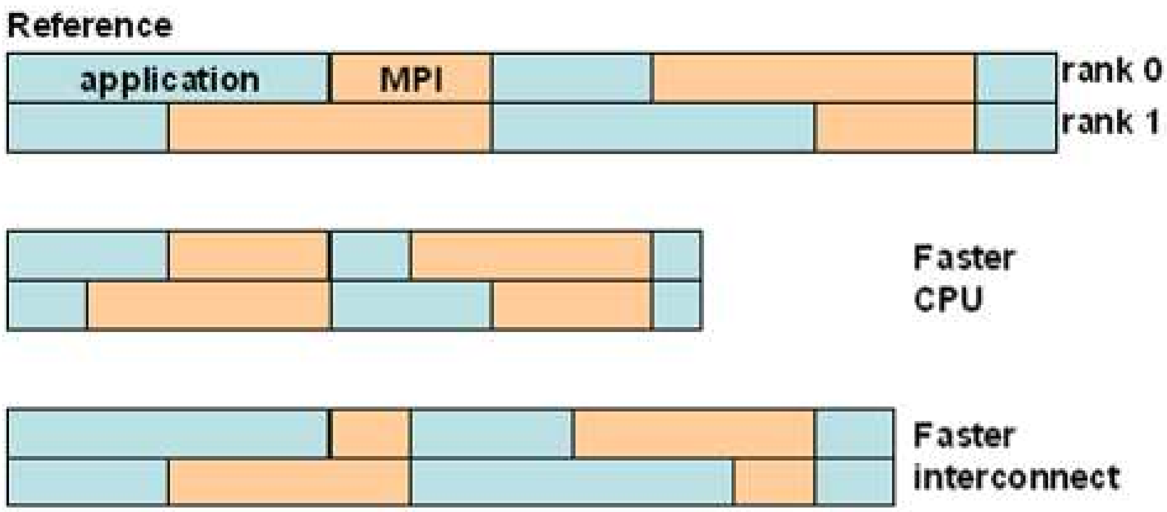
PARATEC Example



Derived Statistics



MPInside Modeling



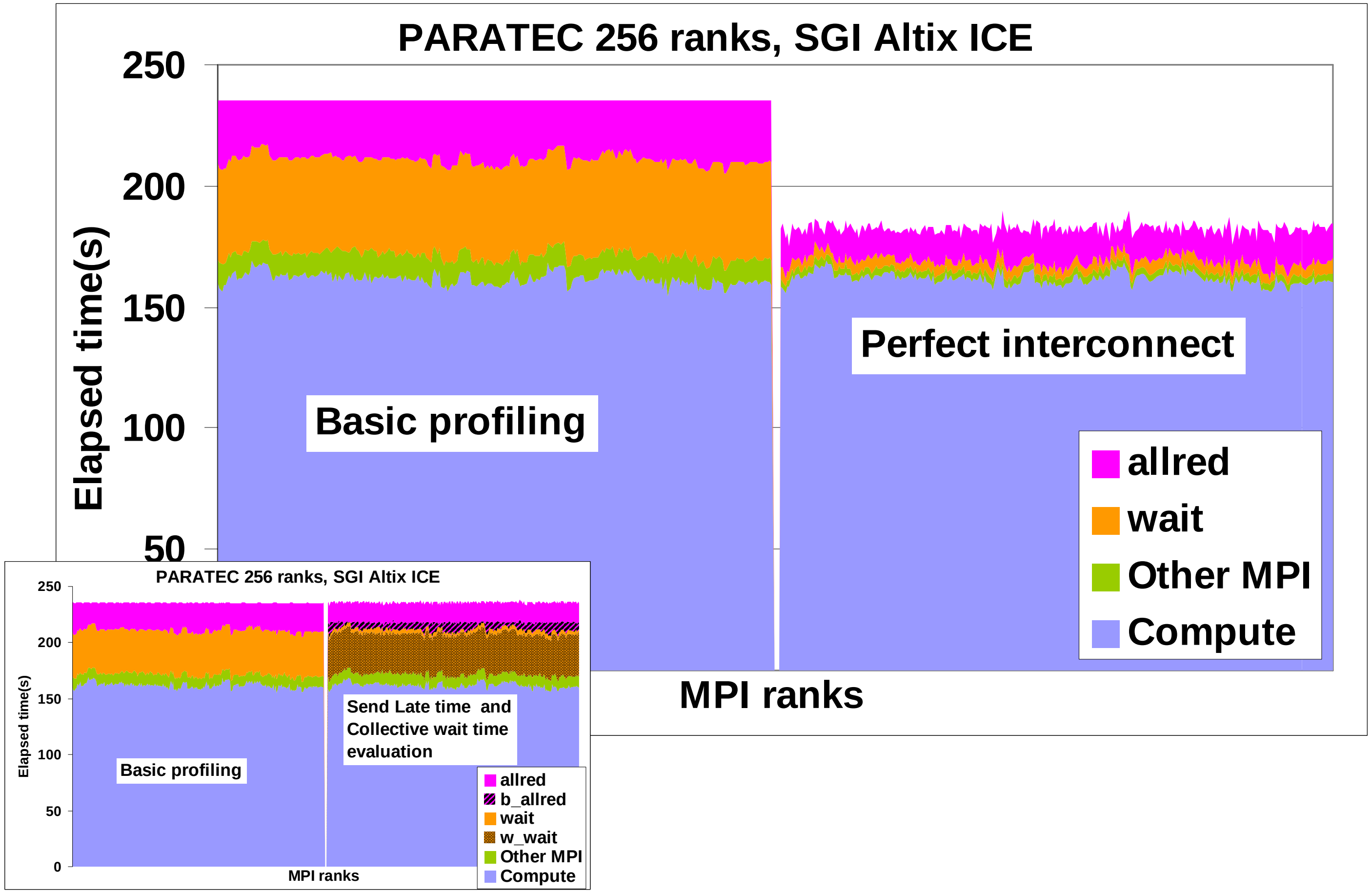
- Uses virtual clocks to perform on-the-fly “what if” experiments. Such virtual clocks are incremented by the measured computational times and by an evaluation of the communication times
- Communication model:
  - T(size) = latency + size / bandwidth(size, network load)
- “Perfect” interconnect:
  - latency = 0, bandwidth = ∞

MPInside Modeling continue

- As there is no standard mechanism in MPI for the library to notify tools for internal event a deep knowledge of the MPI library internals is necessary to handle collective function properly. This is why modeling is restricted to the SGI MPI library

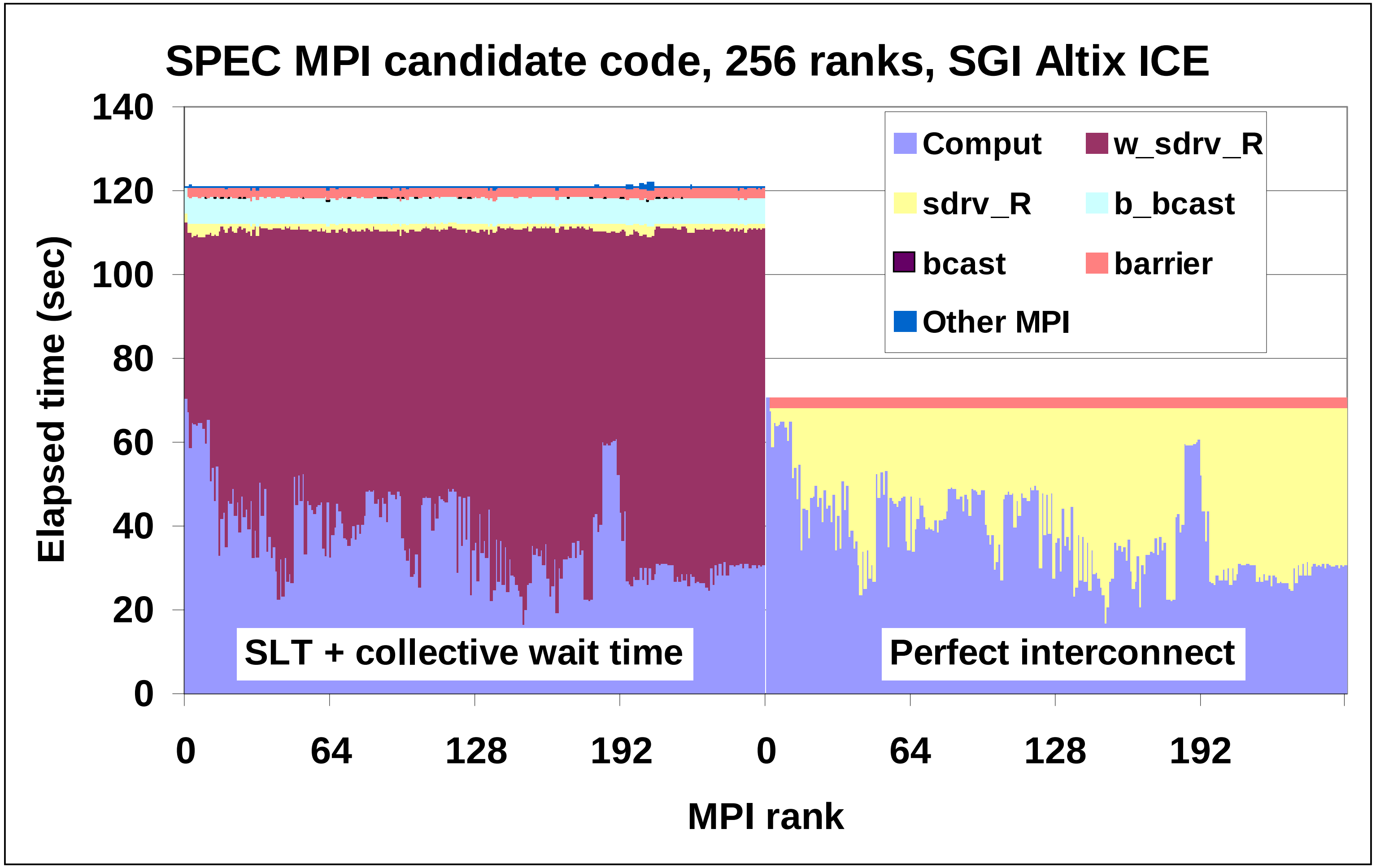
- Perfect interconnect is an exception:
  - For each collective operation all the virtual clocks are exchanged between processors. The latter arrival imposes its clock. Then the collective operation is perfect.

Perfect Interconnect Example



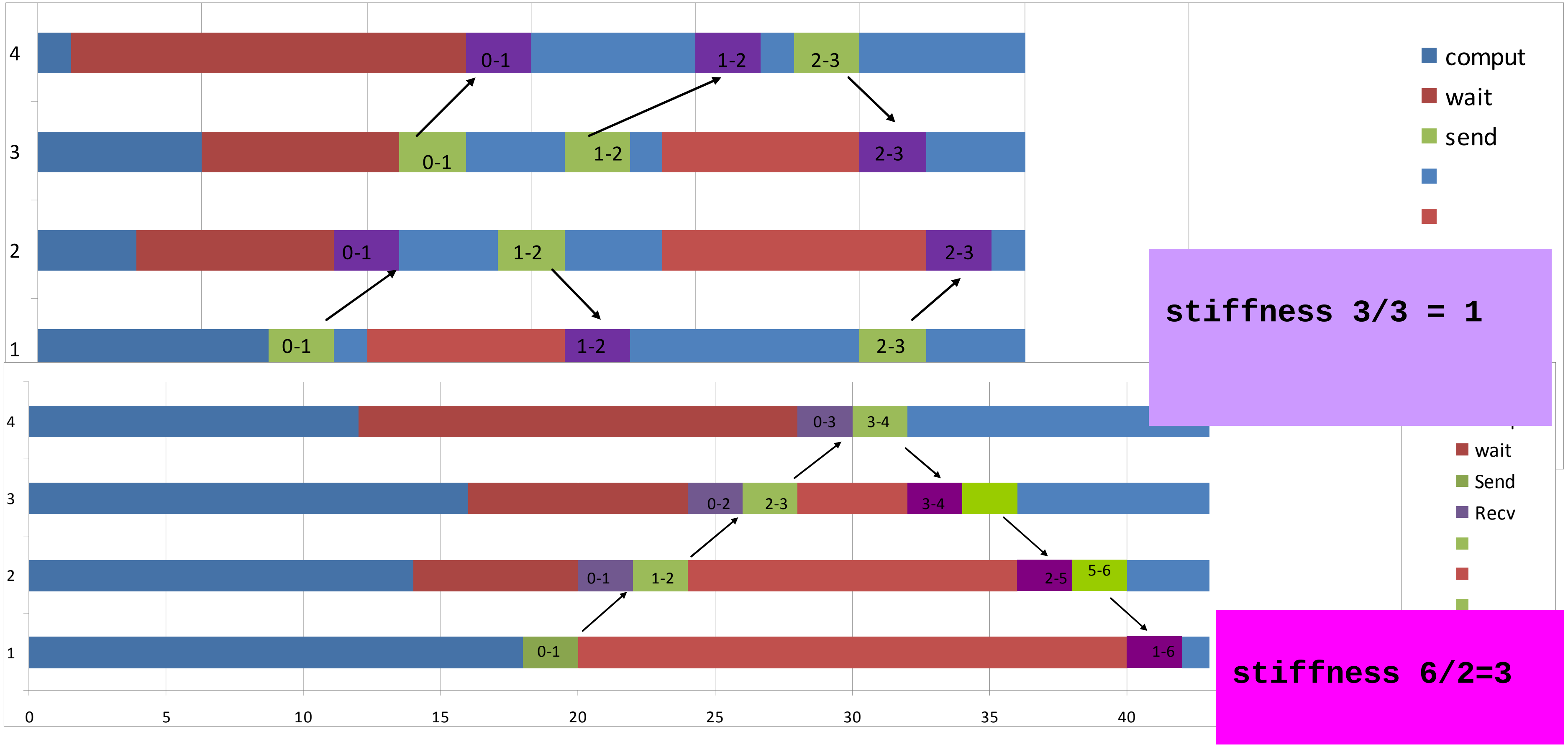
setenv MPINSIDE_MODEL PERFECT+1.0

Case Study



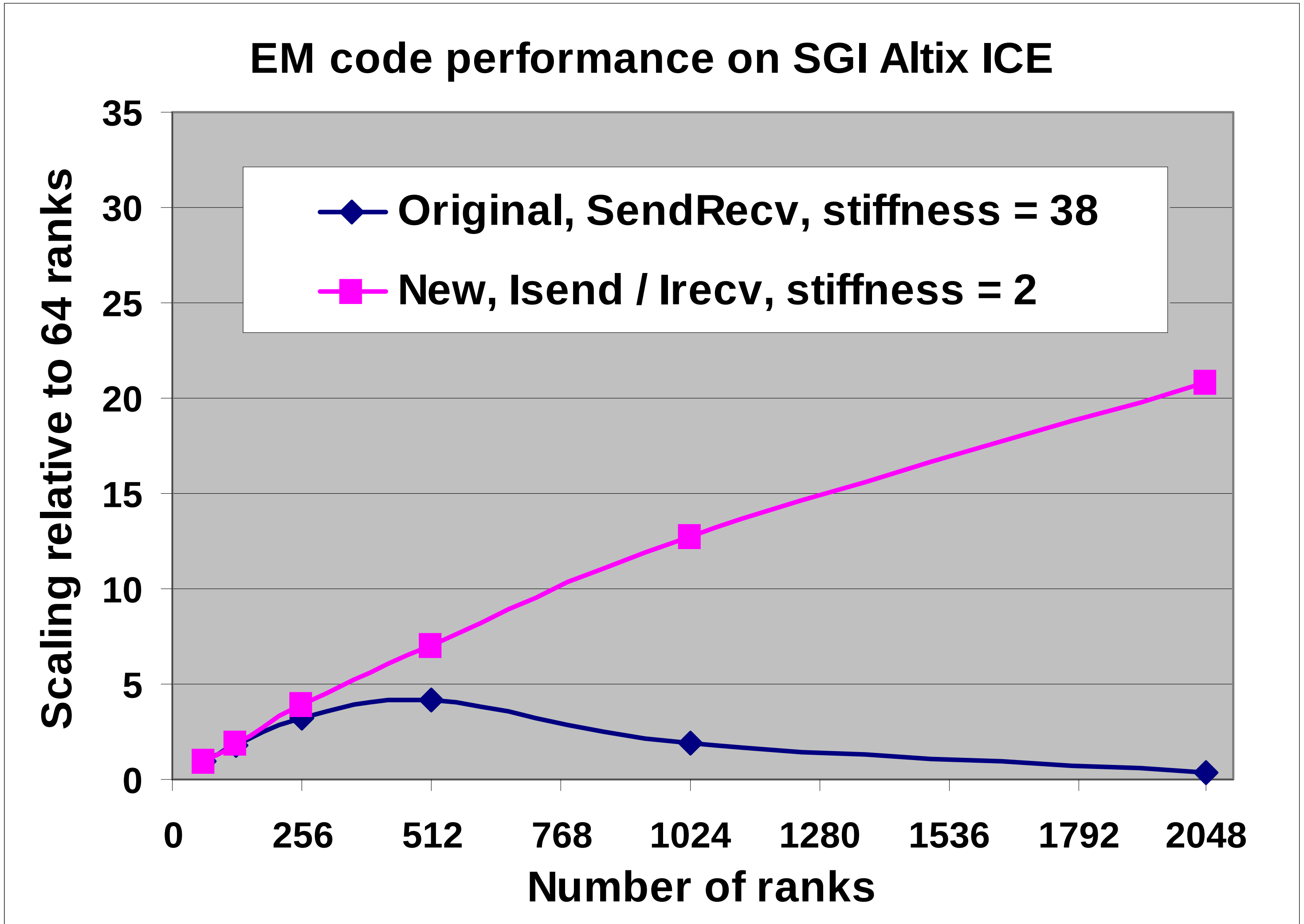
SPEC MPI candidate code, 256 ranks, SGI Altix ICE
Elapsed time (sec)
140
120
100
80
60
40
20
0
Comput
w_sdrv_R
sdrv_R
b_bcast
bcast
barrier
Other MPI
SLT + collective wait time
Perfect interconnect
0
64
128
192
0
64
128
192
MPI rank

Communication "Stiffness"



Such information is carried in the supplemental piece of information necessary for the Send Late Time evaluation

Lowering the Stiffness



MPInside : MPI function “branches” with “partner” cross references

Run:

```
setenv MPINSIDE_CALL_STACK_DEPTH 5
Setenv MPINSIDE_CROSS_REFENCE
mpirun –np xxx MPInside your_apps your_args
```

```
MPInside report rank 0
    MPI_FUNCTION   Brid   Time(s) Self%  Tot% #reqs_S #reqs_R avr_szS  avr_szR W_miss%  Rcv_W(s)
        MPI_Recv   #265     1.552 38.06  38.1       0     105       0  182972   0.0    1.552413
Ancestors: HPL_spreadT HPL_pdlaswp01T HPL_pdupdateTT HPL_pdgesv0 HPL_pdgesv HPL_pdtest
Partners_l_0:   240:#19:14.81:97.18 192:#15:8.91:95.18 96:#15:8.82:95.42 32:#15:7.74:97.41 144:#19:6.53:93.12...
...........
    MPI_FUNCTION   Brid   Time(s) Self%  Tot% #reqs_S #reqs_R avr_szS  avr_szR W_miss%
        MPI_Send   #19     0.125  3.06  90.0     102        0 1370801       0   0.0
Ancestors: HPL_bcast_1rinM HPL_bcast HPL_pdgesv0 HPL_pdgesv HPL_pdtest main
...........
    MPI_FUNCTION   Brid   Time(s) Self%  Tot% #reqs_S #reqs_R avr_szS  avr_szR W_miss%
        MPI_Wait   #524     0.983 24.10  62.2       0    1200       0       0   0.0
Ancestors: HPL_rollT HPL_pdlaswp01T HPL_pdupdateTT HPL_pdgesv0 HPL_pdgesv HPL_pdtest
Partners_l_0:   0:#273:100.00
...........
    MPI_FUNCTION   Brid   Time(s) Self%  Tot% #reqs_S #reqs_R avr_szS  avr_szR W_miss%  Rcv_W(s)
       MPI_Irecv   #273     0.026  0.63  96.9       0    1200       0   70269   0.0    0.982906
Ancestors: HPL_rollT HPL_pdlaswp01T HPL_pdupdateTT HPL_pdgesv0 HPL_pdgesv HPL_pdtest
Partners_l_0:   240:#19:14.30:55.34 208:#19:10.93:56.13 224:#19:9.99:70.97 16:#19:7.91:73.19 160:#16:7.75:48.53...
...........
```

- More about Received partners on next slide.

MPInside :
MPI Receive function branch partners

```
MPI_FUNCTION   Brid    Time(s) Self%  Tot% #reqs_S #reqs_R avr_szS  avr_szR W_miss%  Rcv_W(s)
    MPI_Recv   #265     1.552 38.06  38.1       0     105       0  182972   0.0    1.552413
Ancestors: HPL_spreadT HPL_pdlaswp01T HPL_pdupdateTT HPL_pdgesv0 HPL_pdgesv HPL_pdtest
Partners_l_0:   240:#19:14.81:97.18 192:#15:8.91:95.18 96:#15:8.82:95.42 32:#15:7.74:97.41
```

- Partner list format: **CPU:#Branch:Wait:Send_late**

CPU: Rank number that did an MPI Send/Isend for this branch:
#Branch: MPI_Send/Isend Branch ident(brid):
Wait: percent of this `MPI_Recv` that involved this "A" rank "#B" MPI Send/Isend branch
Send_late: percent of this MPI_Recv where the corresponding Send was arriving late
For example: "240:#19:14.81:97.18" means:
This MPI_Recv branch was "partner" with the MPI_Send branch id 19 of CPU 240 and this partnership is accountable for 14.81% of this MPI_Recv branch communication time and 97.18 % of this 14.81% was just wait because the sends were arriving late

MPInside availability

- Basic profiling functionality is supported for SGI MPT, Intel MPI, HP MPI and ScaliMPI Platform MPI
  - Open MPI support to be added soon.
- General modeling capabilities require detailed knowledge of the inner workings of the library
  - Current support is for SGI MPT only.
- Perfect interconnect modeling is currently supported on all MPI supported. MPInside is available via SupportFolio

SIPEW 2010

sgi

Tools need MPI standardizations

- Performance tools are of great importance for parallel applications in particular for MPI applications.
- MPI Standard only provides the "PMPI" mechanism allowing easy wrapping of MPI functions
  - Better than nothing but this is not much as wrapping is easy
- Advance profiling interfaces should be part of the standard:
  - For notification to the tools about MPI internal library events
    - P2P collective transfers
    - Operation delayed because of lack of buffers
    - ..
  - A mechanism to carry supplemental information that the tools may wish to associate with user messages
- Standard don't want to impose a particular implementation but all the MPI work more or less the same. Based on this experience more attention should be given to MPI tools

accelerating results™