Master thesis

Master in Research and Innovation

Inferring programs structure from an execution trace

Author

Juan Francisco Martínez Vera

Supervisor

Jesús Labarta Mancho

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

Outline for section 1

Performance Analysis

- Aid to detect **bottlenecks**
  - That prevents from better performance
- Requires high skilled analyst...
- ... so developers are not used to work with them
  - But derive this work to actual specialist
  - Analyst are used to **work with codes they are not familiar with**.
  - e.g. POP project: Provides **performance optimisation and productivity** services for academic and industrial code(s).

Outline for section 2

Motivations

About improve understandability

Providing application structure will lead to better understandability about what the application is doing.

Motivations

About improve understandability

Providing application structure will lead to better understandability about what the application is doing.

About improve reports

Having the structure of the application the communication with developer can be improved

Outline for section 3

Objective

The objective for this thesis is...

Expose the structure of the application, by means of a post-mortem trace analysis.

Outline for section 4

State of the Art

First step is always to explore what is over there...

- Some previous research have been driven in this field.
- Since traces are ordered sequences...
- ... Natural choice has been **sequential pattern mining** in general.

State of the Art

First step is always to explore what is over there...

- Some previous research have been driven in this field.
- Since traces are ordered sequences...
- ... Natural choice has been **sequential pattern mining** in general.

Just **pick and develop one of the proposals?**

State of the Art

First step is always to explore what is over there...

- Some previous research have been driven in this field.
- Since traces are ordered sequences...
- ... Natural choice has been **sequential pattern mining** in general.

Just **pick and develop one of the proposals?**

Looking to the trend of increasing trace sizes...

- This sort of algorithms, **used to present high complexity**.
- From $O(n^2)$ to $O(2^n)$.

State of the Art

First step is always to explore what is over there...

- Some previous research have been driven in this field.
- Since traces are ordered sequences...
- ... Natural choice has been **sequential pattern mining** in general.

Just **pick and develop one of the proposals?**

Looking to the trend of increasing trace sizes...

- This sort of algorithms, **used to present high complexity**.
- From $O(n^2)$ to $O(2^n)$.

Finally...

The decision has been to explore a new approach!

Outline for section 5

Application structure by classification

The key idea

Use communications as **proxies** for the observation of iterations and clustering them **by its behaviour**.

Application structure by classification

The key idea

Use communications as **proxies** for the observation of iterations and clustering them **by its behaviour**.

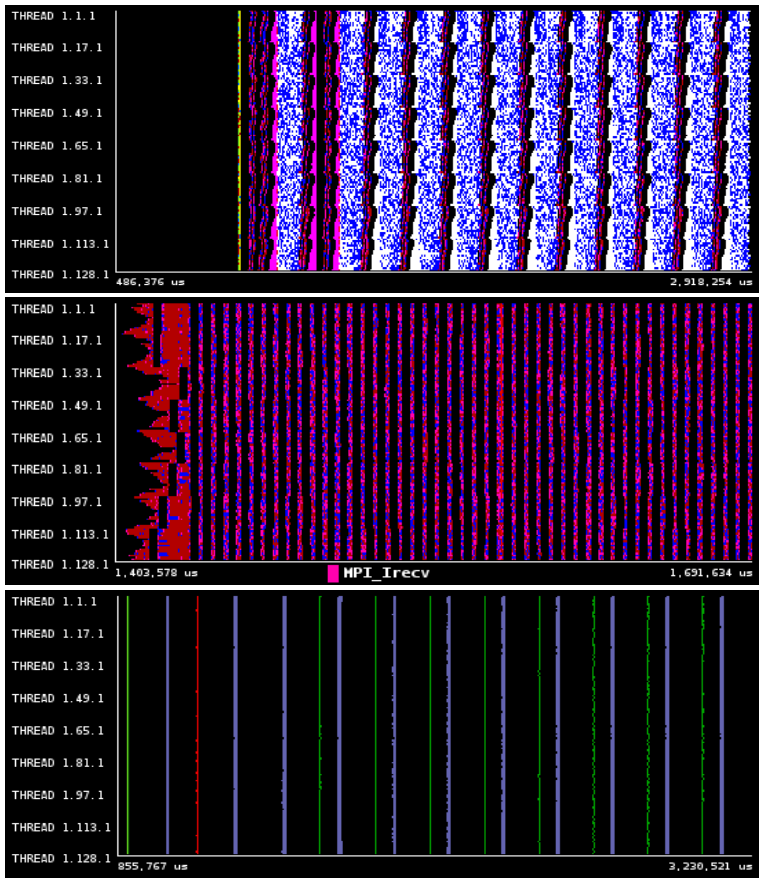
Why?

HPC applications idiosincracy

- Big outer loop
- Repetitive and stable executions
- Communications lies on loops that drives the execution

So...

- **Similar behaviour for all iterations** in a given loop
- Loops by **monitoring the communications**.



Application structure by classification

What will define **behaviour**?
Selected features must be able to

- Join MPIs from the same loop
- Separe MPIs from different loops

Application structure by classification

What will define **behaviour**?
Selected features must be able to

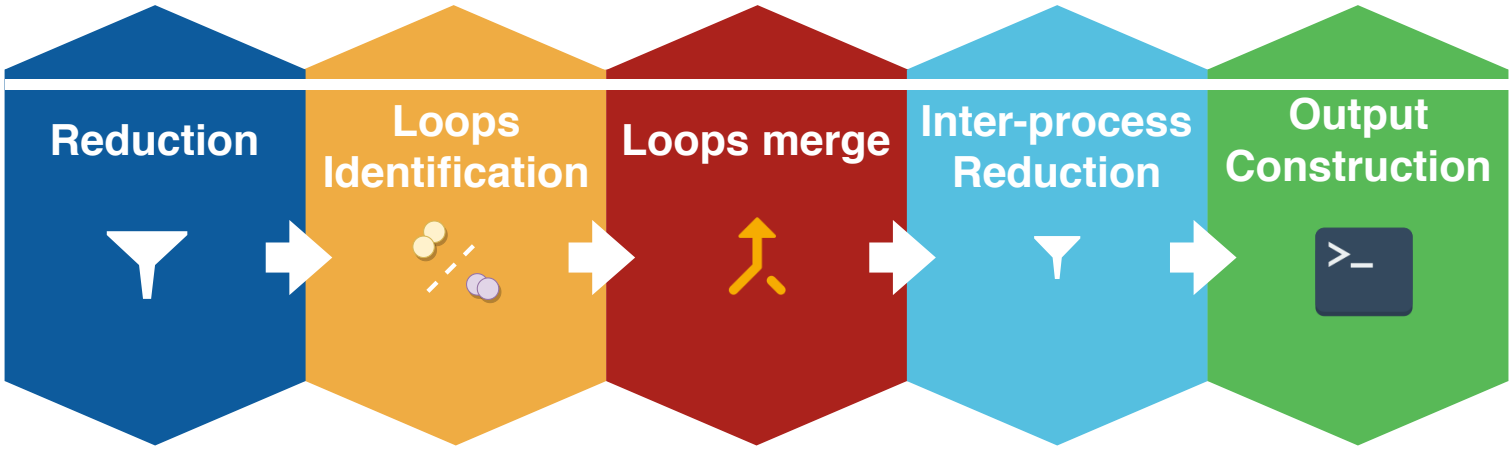
- Join MPIs from the same loop
- Separate MPIs from different loops

As a starting point ...

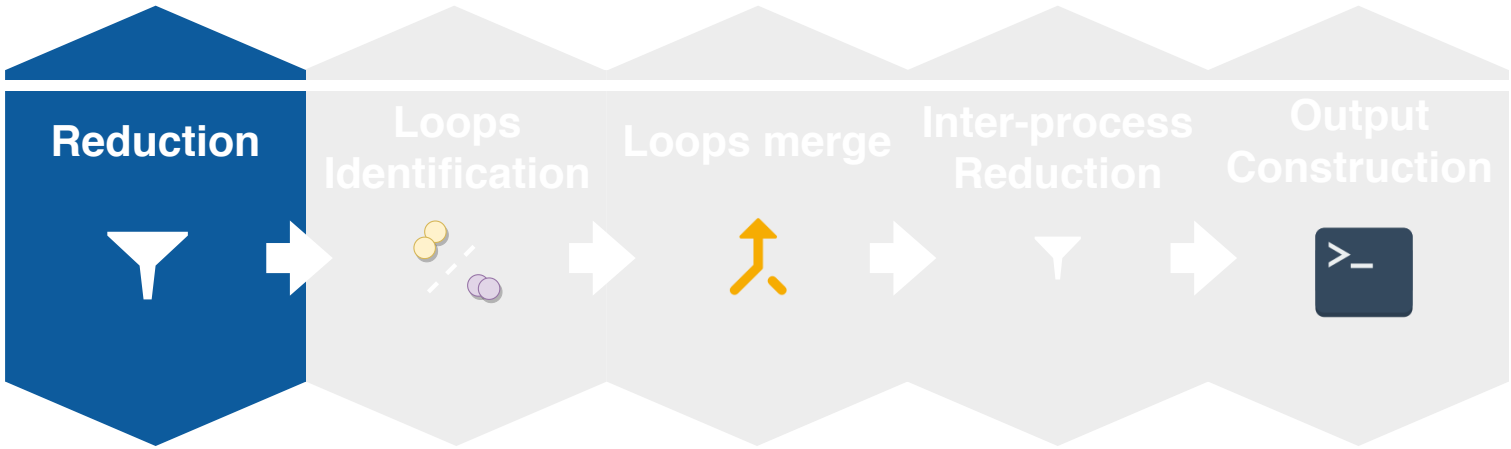
1. **Number of repetitions**: Two different mpi calls in same loop will be executed the same number of times.
2. **Mean time between repetitions**: Two different loops will, presumibly, execute different work.

Outline for section 6

Workflow

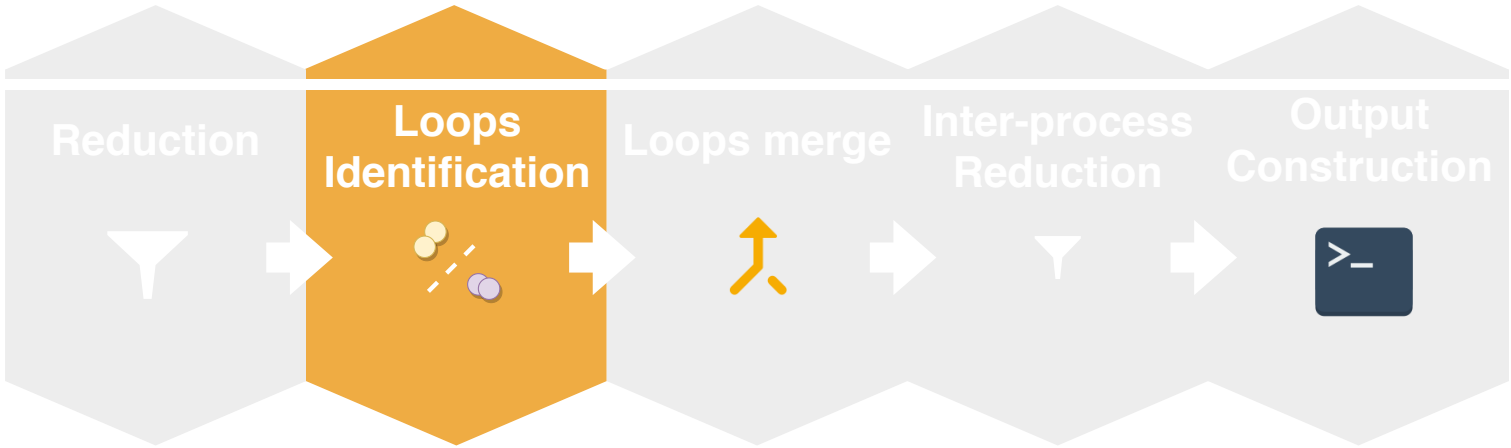

Reduction
Loops Identification
Loops merge
Inter-process Reduction
Output Construction

Workflow



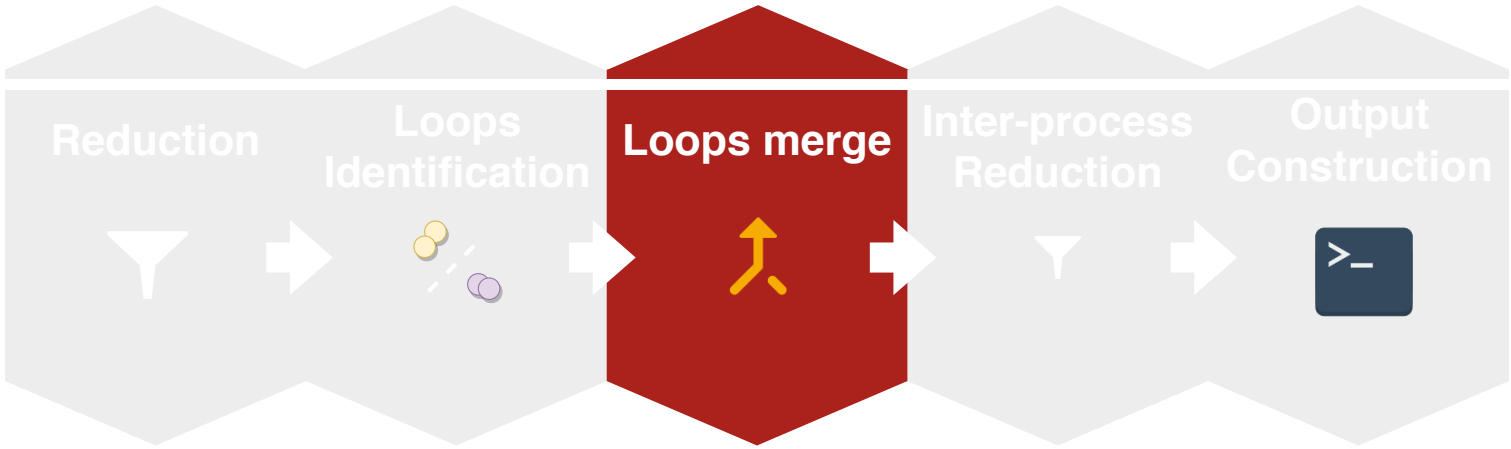
- Parsing and reducing information from trace

Workflow



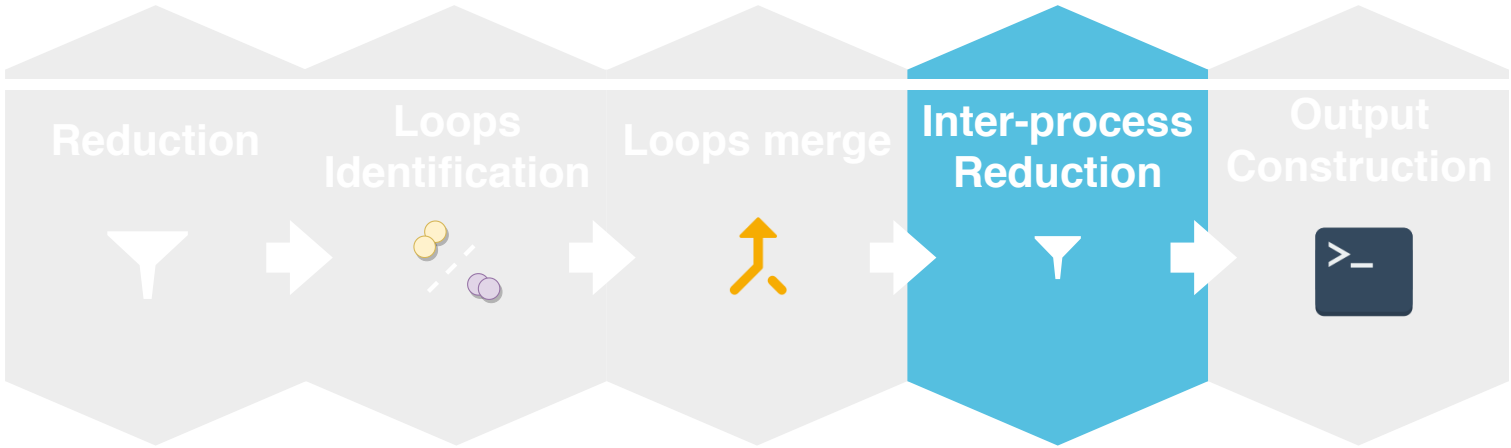
- Parsing and reducing information from trace
- Identify loops by MPI call sites classification

Workflow



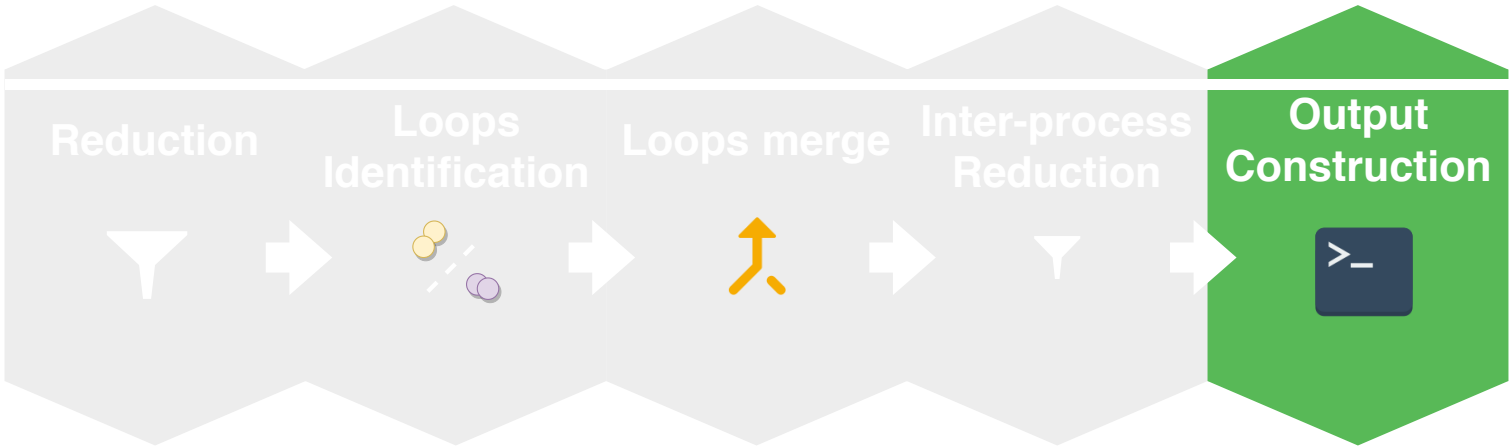
- Parsing and reducing information from trace
- Identify loops by MPI call sites classification
- Find out the hierarchical relations between loops

Workflow



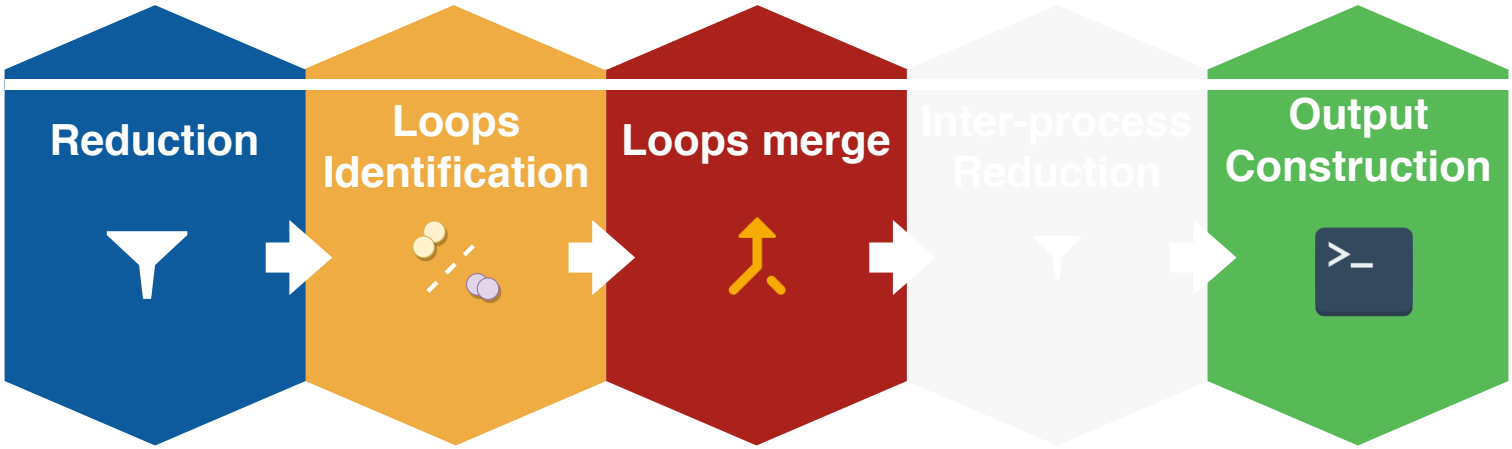
- Parsing and reducing information from trace
- Identify loops by MPI call sites classification
- Find out the hierarchical relations between loops
- Colapse same MPI call site from different processes

Workflow



- Parsing and reducing information from trace
- Identify loops by MPI call sites classification
- Find out the hierarchical relations between loops
- Colapse same MPI call site from different processes
- Constuction of the output

Workflow



- Parsing and reducing information from trace
- Identify loops by MPI call sites classification
- Find out the hierarchical relations between loops
- Colapse same MPI call site from different processes
- Constuction of the output

Workflow

Trace reduction step

Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.1: ()

for 1 to 50
  do
    for 1 to 2
      do { *MPI_A*
    MPI_B
for 1 to 100
  do
    for 1 to 2
      do
        for 1 to 2
          do { *MPI_C*
        MPI_D
    MPI_E

Workflow

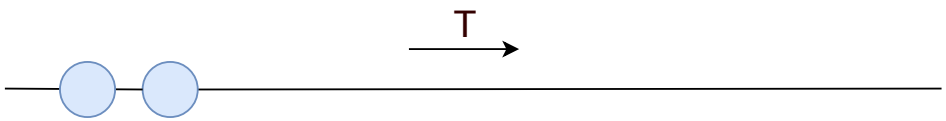
Trace reduction step

Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.2: ()

for 1 to 50
do
  for 1 to 2
  do { *MPI_A*
  MPI_B

for 1 to 100
do
  for 1 to 2
  do
    for 1 to 2
    do { *MPI_C*
    MPI_D
  MPI_E



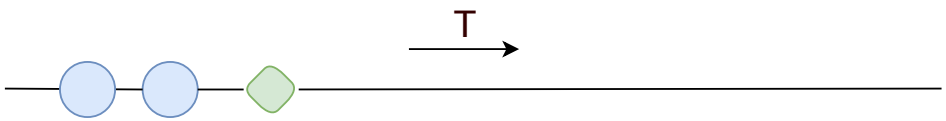
Workflow

Trace reduction step

Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.3: ()

for 1 to 50
do
  for 1 to 2
  do *MPI_A*
  MPI_B
for 1 to 100
do
  for 1 to 2
  do
    for 1 to 2
    do *MPI_C*
    MPI_D
  MPI_E



Workflow

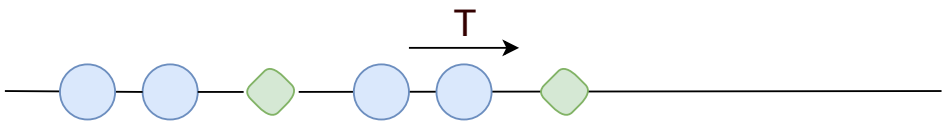
Trace reduction step

Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.4: ()

for 1 to 50
do
  for 1 to 2
  do { *MPI_A*
  MPI_B

for 1 to 100
do
  for 1 to 2
  do
    for 1 to 2
    do { *MPI_C*
    MPI_D
  MPI_E



Workflow

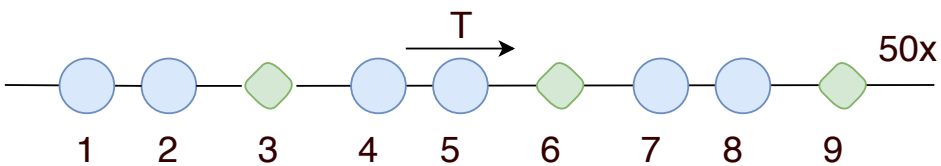
Trace reduction step

Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.5: ()

for 1 to 50
do
  for 1 to 2
  do { *MPI_A*
  MPI_B

for 1 to 100
do
  for 1 to 2
  do
    for 1 to 2
    do { *MPI_C*
    MPI_D
  MPI_E



Workflow

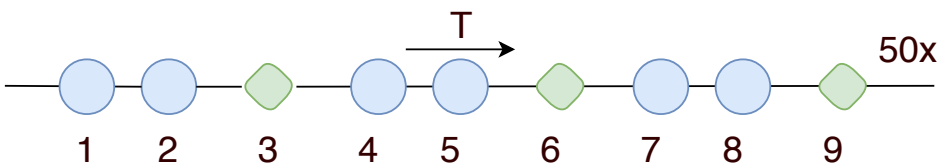
Trace reduction step

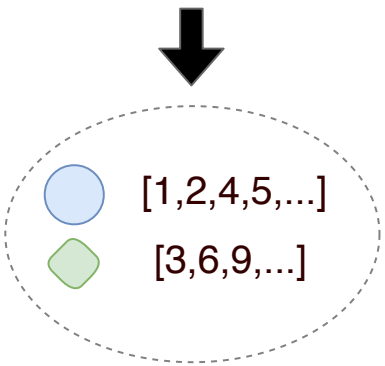
Input Tracefile, i.e. Sequence of timestamped events ordered by time.
Output Set of unique MPI calls sites with attached information.

Algorithm 6.6: ()

for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
  MPI_B

for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
    MPI_D
  MPI_E

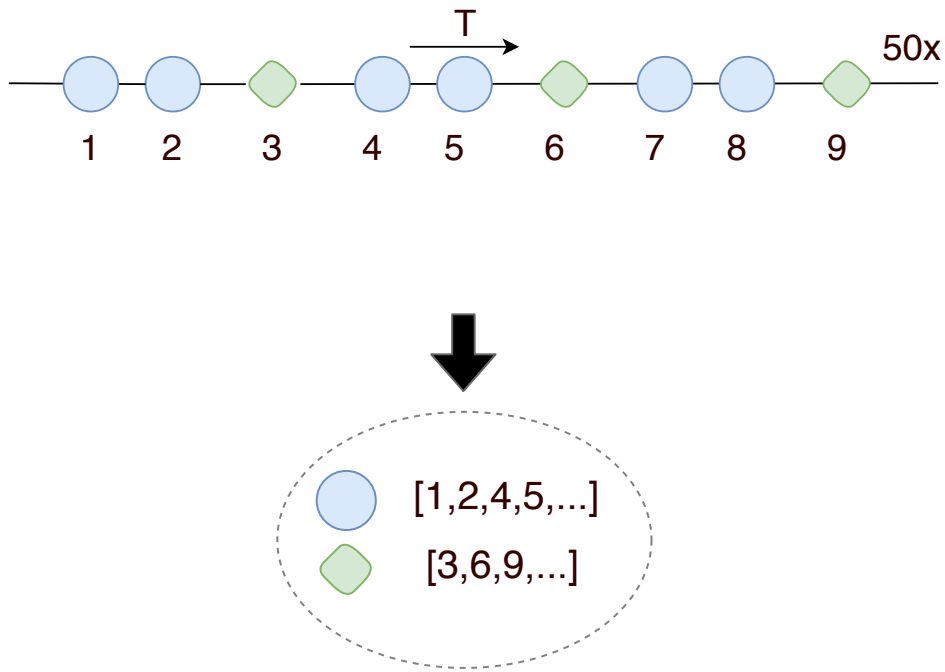




Workflow

Trace reduction step

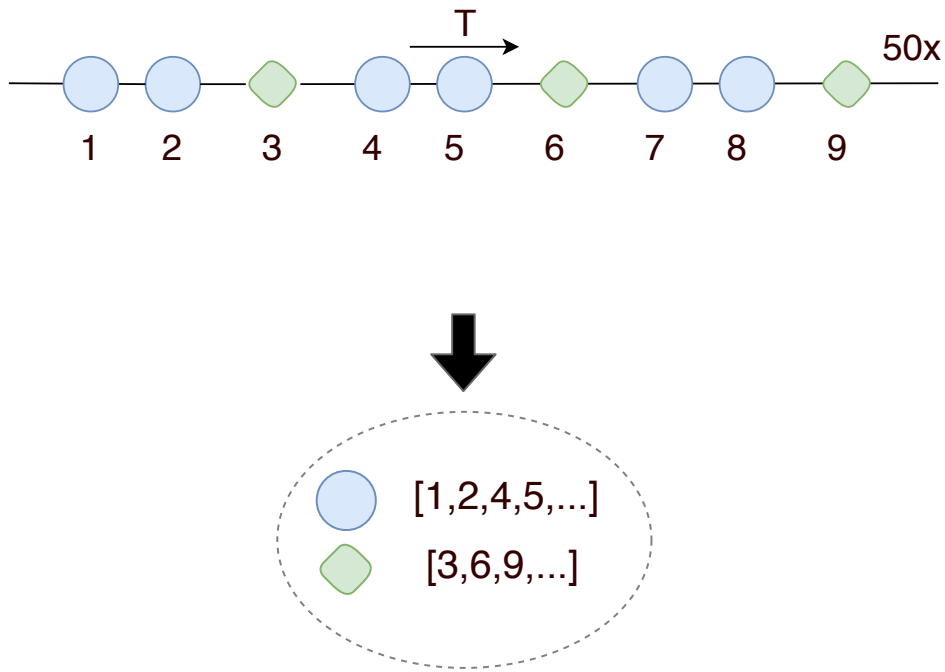
- Is a sort of Map & Reduce
- Every MPI call is identified by its **signature**: Ordered sequence of pairs (*file*, *line*) that define the call path, i.e. The dynamic position
- Additionally **less representative MPI calls are filtered** (10%).



Workflow

Trace reduction step

- Is a sort of Map & Reduce
- Every MPI call is identified by its **signature**: Ordered sequence of pairs (*file*, *line*) that define the call path, i.e. The dynamic position
- Additionally **less representative MPI calls are filtered** (10%).



Keynote

Is **the key step for scalability** because repetitivity of HPC applications

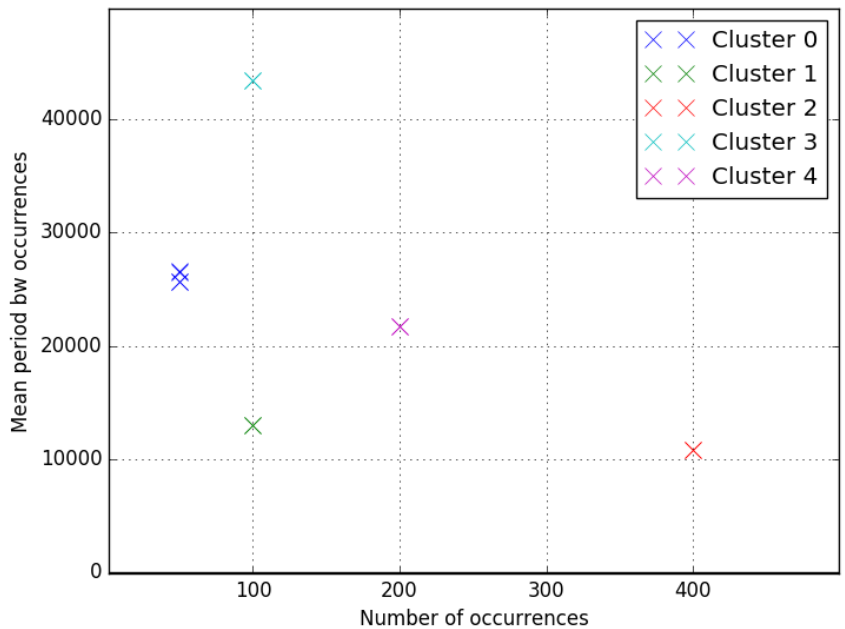
Workflow

Loops identification (ii)

Input Set of unique MPI calls sites with attached information.
Output Set of sets of MPI calls $\rightarrow$ Loops

Algorithm 6.7: ()

for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
  MPI_B
for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
    MPI_D
  MPI_E



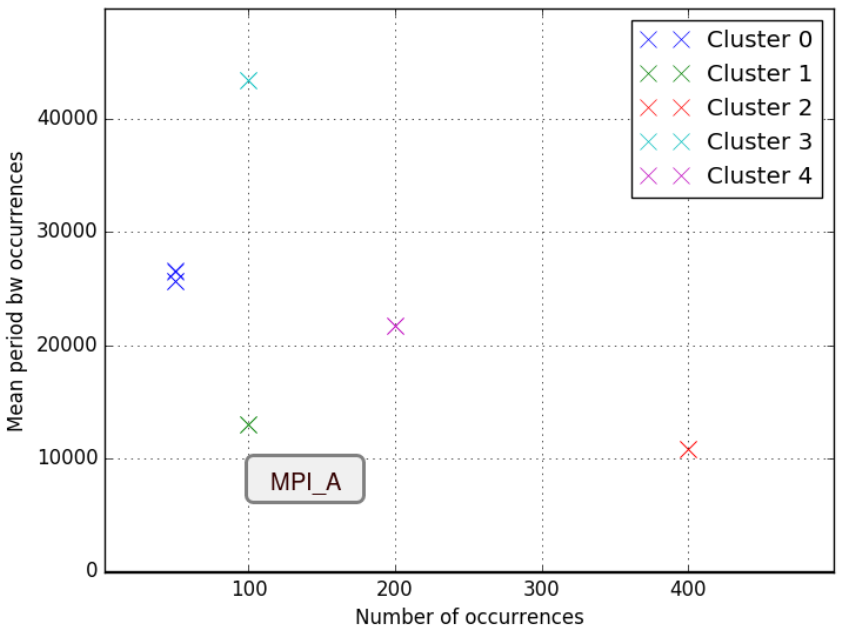
Workflow

Loops identification (ii)

Input Set of unique MPI calls sites with attached information.
Output Set of sets of MPI calls $\rightarrow$ Loops

Algorithm 6.8: ()

for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
  MPI_B
for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
    MPI_D
  MPI_E



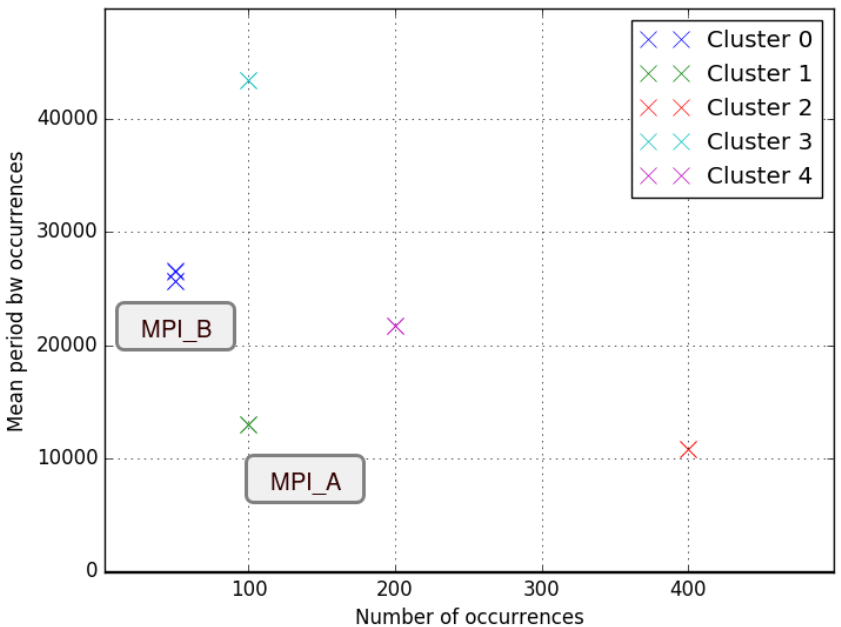
Workflow

Loops identification (ii)

Input Set of unique MPI calls sites with attached information.
Output Set of sets of MPI calls $\rightarrow$ Loops

Algorithm 6.9: ()

for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
  MPI_B
for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
    MPI_D
  MPI_E



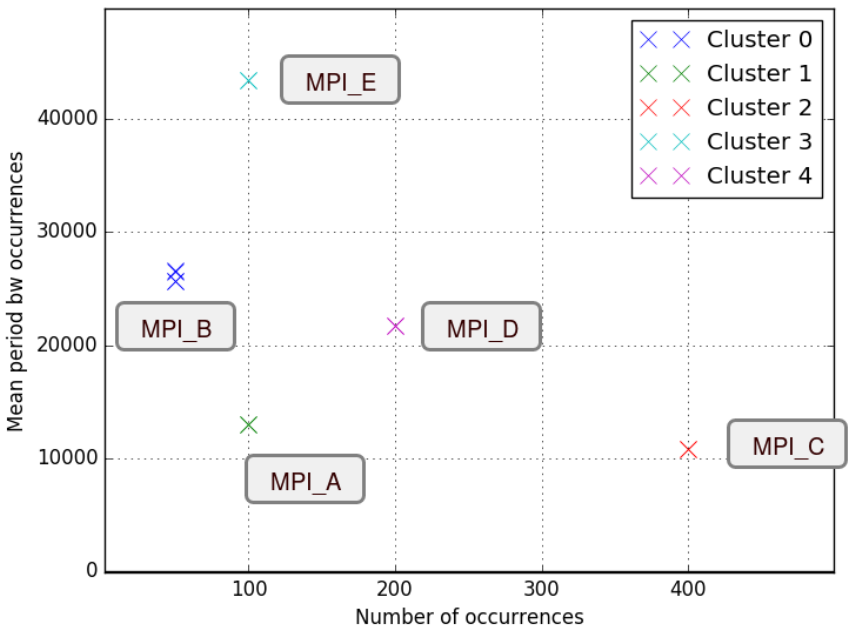
Workflow

Loops identification (ii)

Input Set of unique MPI calls sites with attached information.
Output Set of sets of MPI calls $\rightarrow$ Loops

Algorithm 6.10: ()

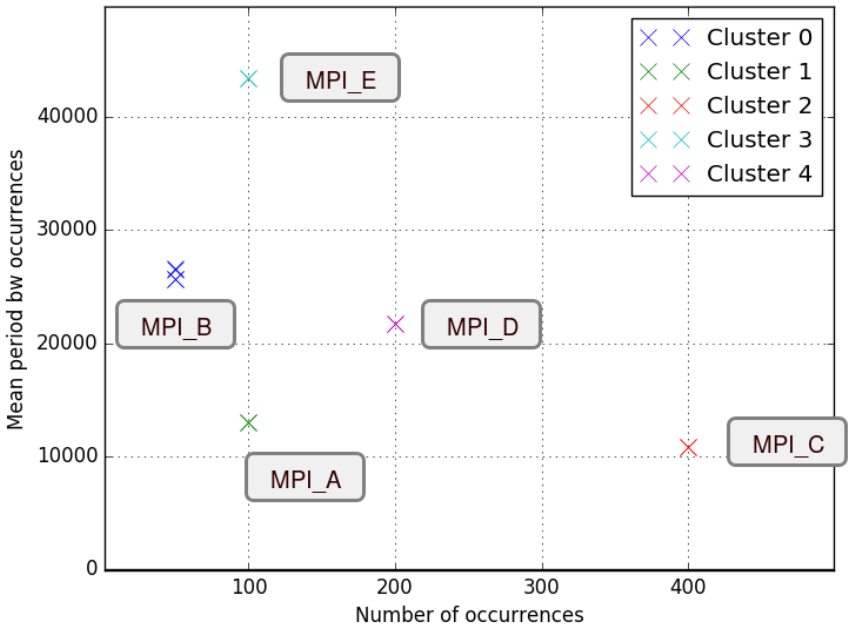
for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
  MPI_B
for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
    MPI_D
  MPI_E



Workflow

Loops identification

- Need for some classification algorithm
- **DBSCAN** as clustering algorithm (prefered to K-means)
  - ϵ empirically set to 0.2 (in general)
  - minPts set to 1
- Resulting clusters **will be considered loops**.
  - Number of repetitions $\rightarrow$ Number of iterations.
  - Mean time between repetitions $\rightarrow$ Mean iterations time.



Workflow

Loops merge (i)

Input Set of loops.
Output Set of top level loops with its related nested loops.

Workflow

Loops merge (i)

Input Set of loops.
Output Set of top level loops with its related nested loops.

Intuition

- Isolated loops are just **pieces of the overall puzzle**.
- By discover its hierarchical relations **the structure of the application will be discovered**...
- As well as the different phases

Workflow

Loops merge (i)

Input Set of loops.
Output Set of top level loops with its related nested loops.

Intuition

- Isolated loops are just **pieces of the overall puzzle**.
- By discover its hierarchical relations **the structure of the application will be discovered**...
- As well as the different phases

We can assume

- Outer loop will have **less iterations** than nested one.
- Outer loop will spend **more time** per iteration.

Workflow

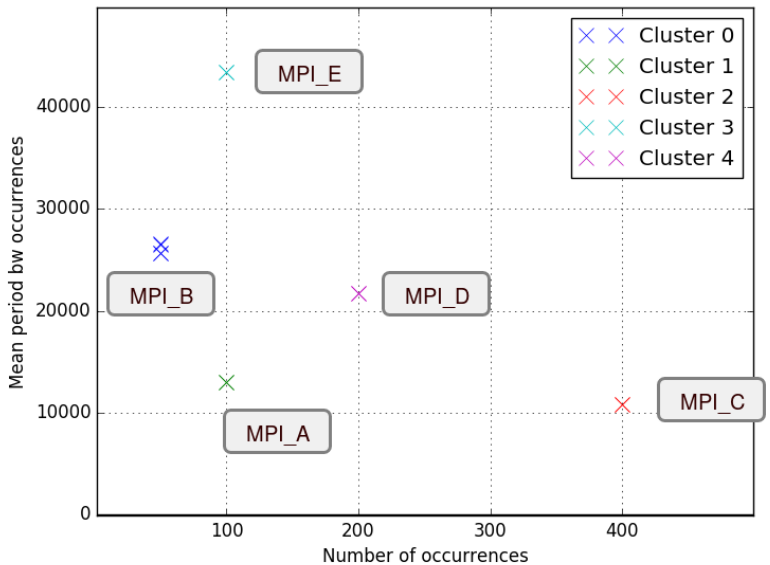
Loops merge (ii)

for 1 to 50

do { **for** 1 to 2 **do** { *MPI_A* ; *MPI_B* }

for 1 to 100

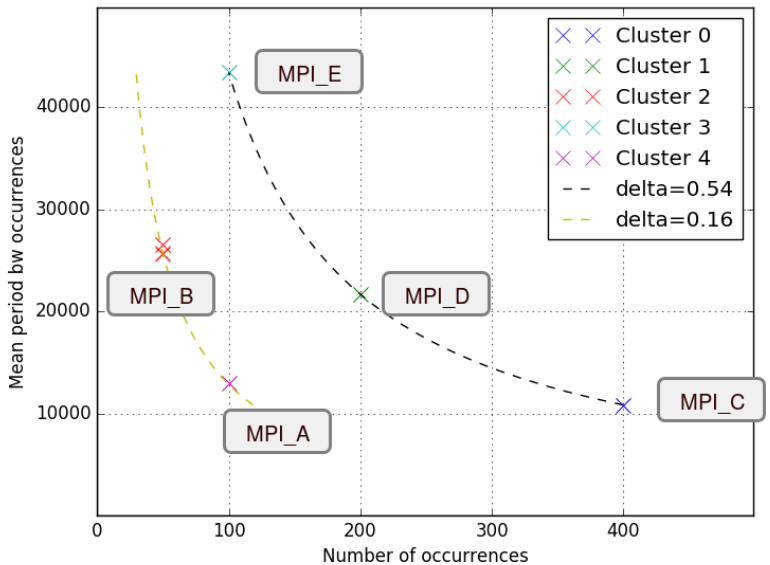
do { **for** 1 to 2 **do** { **for** 1 to 2 **do** { *MPI_C* ; *MPI_D* } } ; *MPI_E* }



Workflow

Loops merge (ii)

for 1 to 50
do {
  for 1 to 2
  do { *MPI_A*
  MPI_B
for 1 to 100
do {
  for 1 to 2
  do {
    for 1 to 2
    do { *MPI_C*
    MPI_D
  MPI_E

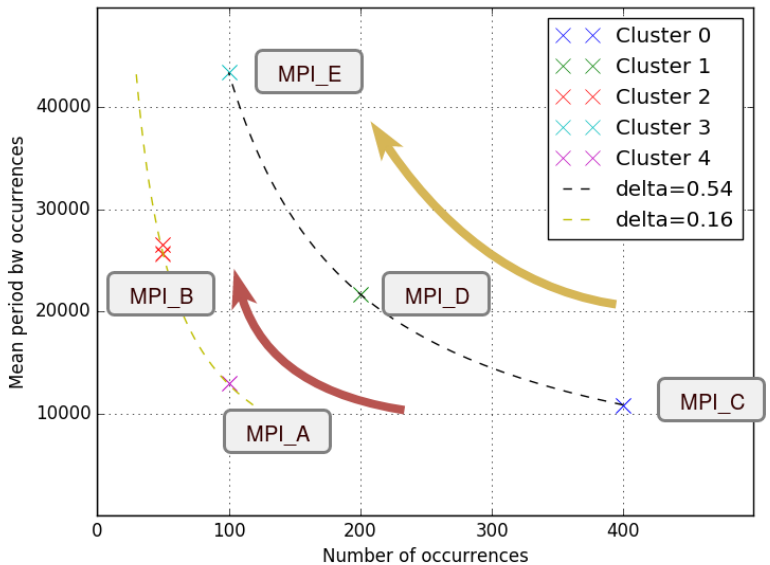


- Classify loops by its delta
  - All lies on same $f(x) = \frac{\delta * T_{exe}}{x}$ being $0 < \delta < 1$
- And then merge them in a less to more iterations fashion
  - Always checking iterations interleaving

Workflow

Loops merge (ii)

for 1 to 50
do {
  for 1 to 2
  do {
    MPI_A
    MPI_B
  }
for 1 to 100
do {
  for 1 to 2
  do {
    for 1 to 2
    do {
      MPI_C
      MPI_D
    }
    MPI_E
  }

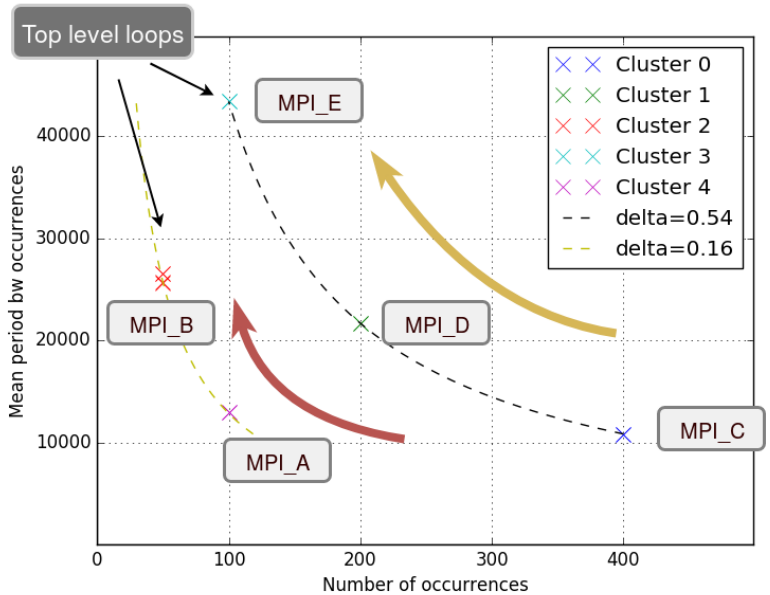


- Classify loops by its delta
  - All lies on same $f(x) = \frac{\delta * T_{exe}}{x}$ being $0 < \delta < 1$
- And then merge them in a less to more iterations fashion
  - Always checking iterations interleaving

Workflow

Loops merge (ii)

for 1 to 50
  do { **for** 1 to 2
    do { *MPI_A*
    MPI_B
for 1 to 100
  do { **for** 1 to 2
    do { **for** 1 to 2
      do { *MPI_C*
      MPI_D
    MPI_E



- Classify loops by its delta
  - All lies on same $f(x) = \frac{\delta * T_{exe}}{x}$ being $0 < \delta < 1$
- And then merge them in a less to more iterations fashion
  - Always checking iterations interleaving

Keynote

This mechanism is then used for phases detection.

Workflow

Pseudocode construction (i)

Input Set of top level loops with rank conditional structures.
Output Pseudocode representing the actual application structure.

Workflow

Pseudocode construction (i)

Input Set of top level loops with rank conditional structures.
Output Pseudocode representing the actual application structure.

- Straightforward construction
- Just a refination of the data is needed
  1. Extracting common call path levels from code block (loops and conditional blocks)
  2. Removing **contiguous repetitive information** what has not been removed in previous step

Workflow

Pseudocode construction (ii)

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 50	-	-	-
test-13.c	17	: : MPI_Barrier(CommId:1)	26.43us	-	0.3
		: : FOR 1 TO 2.0	-	-	-
		: : : IF rank in [1]	-	-	-
test-13.c	23	: : : : MPI_Send(1:0)	22.8us	4.0B	0.53
		: : : IF rank in [0]	-	-	-
test-13.c	25	: : : : MPI_Recv()	27.09us	0.12B	0.54
		: : END LOOP	-	-	-
test-13.c	**28**	**: : [~ computation ~]**	**10.18ms**	**-**	**0.06**
test-13.c	28	: : MPI_Barrier(CommId:1)	106.13us	-	1.43
		: END LOOP	-	-	-
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : FOR 1 TO 2.0	-	-	-
		: : : FOR 1 TO 2.0	-	-	-
		: : : : IF rank in [1]	-	-	-
test-13.c	40	: : : : : MPI_Send(1:0)	23.0us	4.0B	0.5
		: : : : IF rank in [0]	-	-	-
test-13.c	42	: : : : : MPI_Recv()	25.1us	0.05B	0.51
		: : : END LOOP	-	-	-
test-13.c	45	: : : MPI_Barrier(CommId:1)	23.93us	-	0.84
		: : END LOOP	-	-	-
test-13.c	**48**	**: : [~ computation ~]**	**50.2ms**	**-**	**0.07**
test-13.c	48	: : MPI_Barrier(CommId:1)	124.99us	-	1.62
		: END LOOP	-	-	-

Workflow

Pseudocode construction (ii)

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 50	-	-	-
test-13.c	17	: : MPI_Barrier(CommId:1)	26.43us	-	0.3
		: : FOR 1 TO 2.0	-	-	-
		: : : IF rank in [1]	-	-	-
test-13.c	23	: : : : MPI_Send(1:0)	22.8us	4.0B	0.53
		: : : IF rank in [0]	-	-	-
test-13.c	25	: : : : MPI_Recv()	27.09us	0.12B	0.54
		: : END LOOP	-	-	-
test-13.c	**28**	**: : [~ computation ~]**	**10.18ms**	**-**	**0.06**
test-13.c	28	: : MPI_Barrier(CommId:1)	106.13us	-	1.43
		: END LOOP	-	-	-
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : FOR 1 TO 2.0	-	-	-
		: : : FOR 1 TO 2.0	-	-	-
		: : : : IF rank in [1]	-	-	-
test-13.c	40	: : : : : MPI_Send(1:0)	23.0us	4.0B	0.5
		: : : : IF rank in [0]	-	-	-
test-13.c	42	: : : : : MPI_Recv()	25.1us	0.05B	0.51
		: : : END LOOP	-	-	-
test-13.c	45	: : : MPI_Barrier(CommId:1)	23.93us	-	0.84
		: : END LOOP	-	-	-
test-13.c	**48**	**: : [~ computation ~]**	**50.2ms**	**-**	**0.07**
test-13.c	48	: : MPI_Barrier(CommId:1)	124.99us	-	1.62
		: END LOOP	-	-	-

Workflow

Pseudocode construction (ii)

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 50	-	-	-
test-13.c	17	: : MPI_Barrier(CommId:1)	26.43us	-	0.3
		: : FOR 1 TO 2.0	-	-	-
		: : : IF rank in [1]	-	-	-
test-13.c	23	: : : : MPI_Send(1:0)	22.8us	4.0B	0.53
		: : : IF rank in [0]	-	-	-
test-13.c	25	: : : : MPI_Recv()	27.09us	0.12B	0.54
		: : END LOOP	-	-	-
test-13.c	**28**	**: : [~ computation ~]**	**10.18ms**	**-**	**0.06**
test-13.c	28	: : MPI_Barrier(CommId:1)	106.13us	-	1.43
		: END LOOP	-	-	-
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : FOR 1 TO 2.0	-	-	-
		: : : FOR 1 TO 2.0	-	-	-
		: : : : IF rank in [1]	-	-	-
test-13.c	40	: : : : : MPI_Send(1:0)	23.0us	4.0B	0.5
		: : : : IF rank in [0]	-	-	-
test-13.c	42	: : : : : MPI_Recv()	25.1us	0.05B	0.51
		: : : END LOOP	-	-	-
test-13.c	45	: : : MPI_Barrier(CommId:1)	23.93us	-	0.84
		: : END LOOP	-	-	-
test-13.c	**48**	**: : [~ computation ~]**	**50.2ms**	**-**	**0.07**
test-13.c	48	: : MPI_Barrier(CommId:1)	124.99us	-	1.62
		: END LOOP	-	-	-

Workflow

Pseudocode construction (ii)

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 50	-	-	-
test-13.c	17	: : MPI_Barrier(CommId:1)	26.43us	-	0.3
		: : FOR 1 TO 2.0	-	-	-
		: : : IF rank in [1]	-	-	-
test-13.c	23	: : : : MPI_Send(1:0)	22.8us	4.0B	0.53
		: : : IF rank in [0]	-	-	-
test-13.c	25	: : : : MPI_Recv()	27.09us	0.12B	0.54
		: : END LOOP	-	-	-
test-13.c	**28**	**: : [~ computation ~]**	**10.18ms**	**-**	**0.06**
test-13.c	28	: : MPI_Barrier(CommId:1)	106.13us	-	1.43
		: END LOOP	-	-	-
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : FOR 1 TO 2.0	-	-	-
		: : : FOR 1 TO 2.0	-	-	-
		: : : : IF rank in [1]	-	-	-
test-13.c	40	: : : : : MPI_Send(1:0)	23.0us	4.0B	0.5
		: : : : IF rank in [0]	-	-	-
test-13.c	42	: : : : : MPI_Recv()	25.1us	0.05B	0.51
		: : : END LOOP	-	-	-
test-13.c	45	: : : MPI_Barrier(CommId:1)	23.93us	-	0.84
		: : END LOOP	-	-	-
test-13.c	**48**	**: : [~ computation ~]**	**50.2ms**	**-**	**0.07**
test-13.c	48	: : MPI_Barrier(CommId:1)	124.99us	-	1.62
		: END LOOP	-	-	-

Workflow

Pseudocode construction (ii)

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 50	-	-	-
test-13.c	17	: : MPI_Barrier(CommId:1)	26.43us	-	0.3
		: : FOR 1 TO 2.0	-	-	-
		: : : IF rank in [1]	-	-	-
test-13.c	23	: : : : MPI_Send(1:0)	22.8us	4.0B	0.53
		: : : IF rank in [0]	-	-	-
test-13.c	25	: : : : MPI_Recv()	27.09us	0.12B	0.54
		: : END LOOP	-	-	-
test-13.c	**28**	**: : [~ computation ~]**	**10.18ms**	**-**	**0.06**
test-13.c	28	: : MPI_Barrier(CommId:1)	106.13us	-	1.43
		: END LOOP	-	-	-
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : FOR 1 TO 2.0	-	-	-
		: : : FOR 1 TO 2.0	-	-	-
		: : : : IF rank in [1]	-	-	-
test-13.c	40	: : : : : MPI_Send(1:0)	23.0us	4.0B	0.5
		: : : : IF rank in [0]	-	-	-
test-13.c	42	: : : : : MPI_Recv()	25.1us	0.05B	0.51
		: : : END LOOP	-	-	-
test-13.c	45	: : : MPI_Barrier(CommId:1)	23.93us	-	0.84
		: : END LOOP	-	-	-
test-13.c	**48**	**: : [~ computation ~]**	**50.2ms**	**-**	**0.07**
test-13.c	48	: : MPI_Barrier(CommId:1)	124.99us	-	1.62
		: END LOOP	-	-	-

```
Welcome to structure detection tool interactive shell
Barcelona Supercomputing Center - Centro nacional de Supercomputacion
Performance tools team - tools@bsc.es

(struct_detection)> help

Documented commands (type help <topic>):
========================================
clustering  help  paraver  pseudocode  q  quit

(struct_detection)> help pseudocode
pseudocode commands:
wo-cs: Do not show call paths
w-burst-info: Show CPU burst information
w-burst-threshold: Show burst above thresshold
default: Default information
ranks: Show pseudocode just for rank
```

Outline for section 7

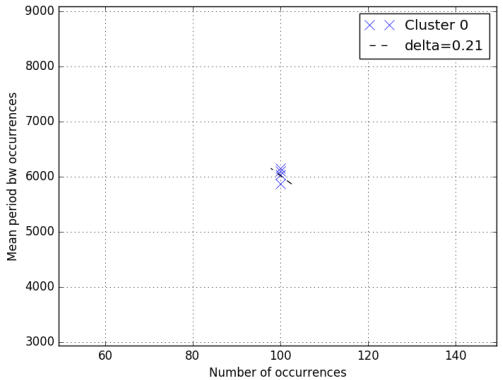
Best features analysis

Motivation (i)

Clustering aliasing $\rightarrow$ Same behavior

$$\textbf{for } 1 to 100 \; \textbf{do} \begin{cases} MPI\_A() \\ MPI\_B() \end{cases}$$

$$\textbf{for } 1 to 100 \; \textbf{do} \begin{cases} MPI\_C() \\ MPI\_D() \end{cases}$$

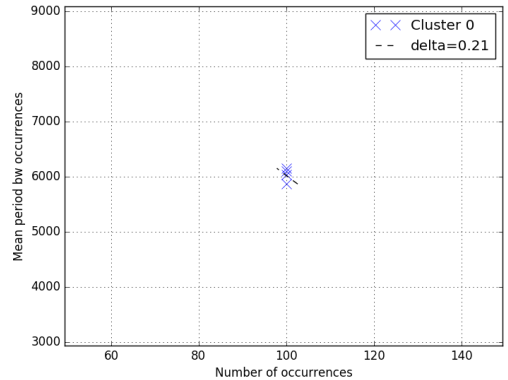


Best features analysis

Motivation (i)

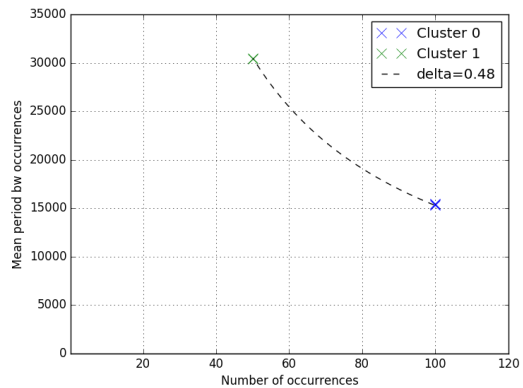
Clustering aliasing $\rightarrow$ Same behavior

$$\begin{array}{l} \textbf{for } 1 to 100 \\ \quad \textbf{do } \begin{cases} MPI\_A() \\ MPI\_B() \end{cases} \\ \textbf{for } 1 to 100 \\ \quad \textbf{do } \begin{cases} MPI\_C() \\ MPI\_D() \end{cases} \end{array}$$



Clustering split $\rightarrow$ Data conditions

$$\begin{array}{l} \textbf{for } i = 1 \text{ to } 10 \\ \quad \textbf{do } \begin{cases} MPI\_B() \\ \textbf{if } isPair(i) \\ \quad \textbf{then } MPI\_A() \\ MPI\_B() \end{cases} \end{array}$$



Best features analysis

Motivation (ii)

Both **solved**, but better if **avoid them**, so...

- **Validate** our first intutition
- **Explore** alternative/additional features
  - IPC, %Loads, %(Cond, Uncond, Total) branches [1]

[1] Strictly limited by available hardaware

Best features analysis

Motivation (ii)

Both **solved**, but better if **avoid them**, so...

- **Validate** our first intutition
- **Explore** alternative/additional features
  - IPC, %Loads, %(Cond, Uncond, Total) branches [1]

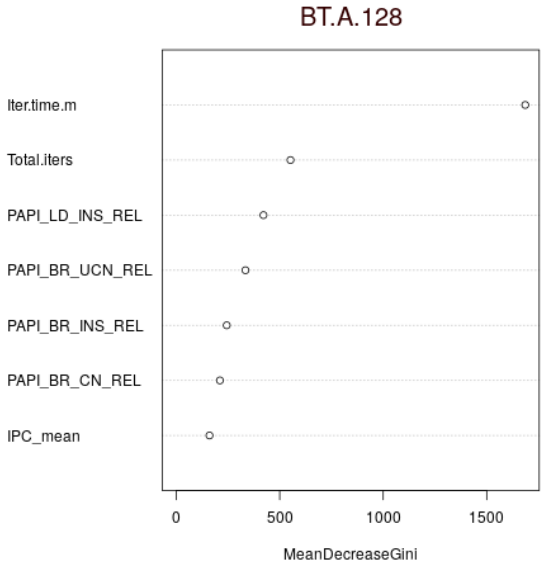
Two steps:

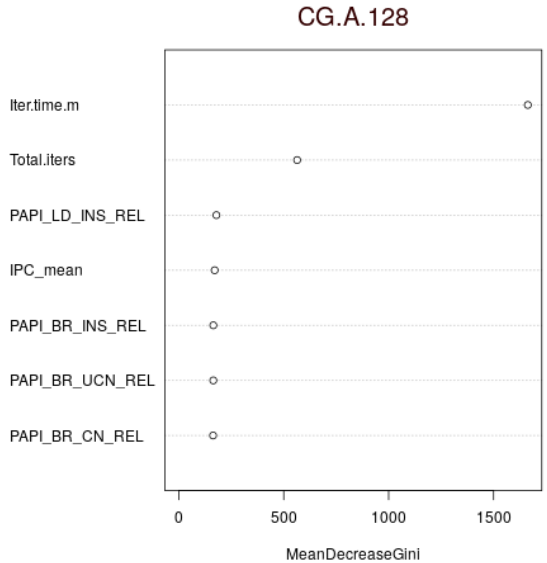
1. Data acquisition
   - Extrae just can monitorize calls to shared libraries
   - Loops monitors injection by **Mercurium**
2. Analyze results
   - Principal Components Analysis
   - Random Forest Variable Importance

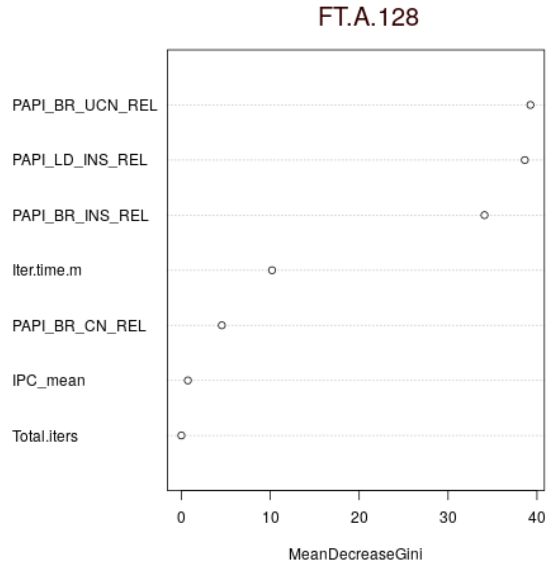
[1] Strictly limited by available hardaware

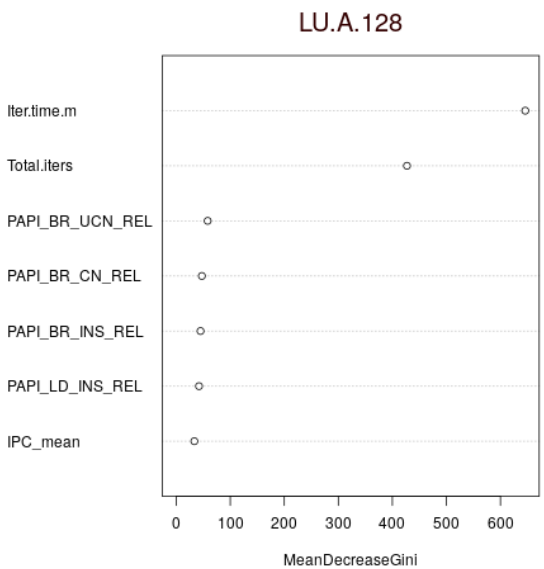
Best features analysis

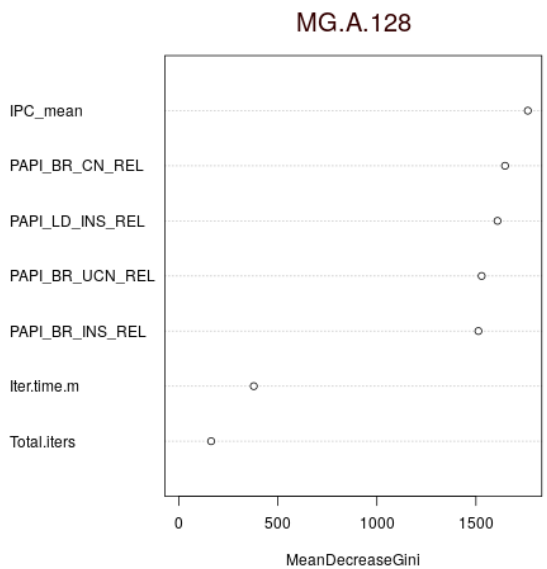
Analysis of results: **Variable Importance**

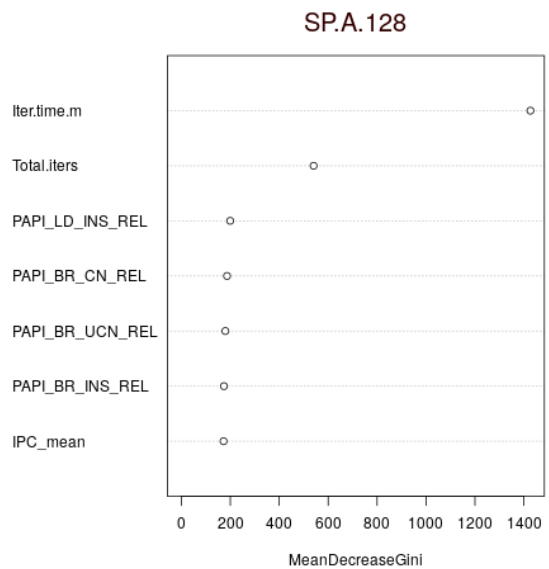






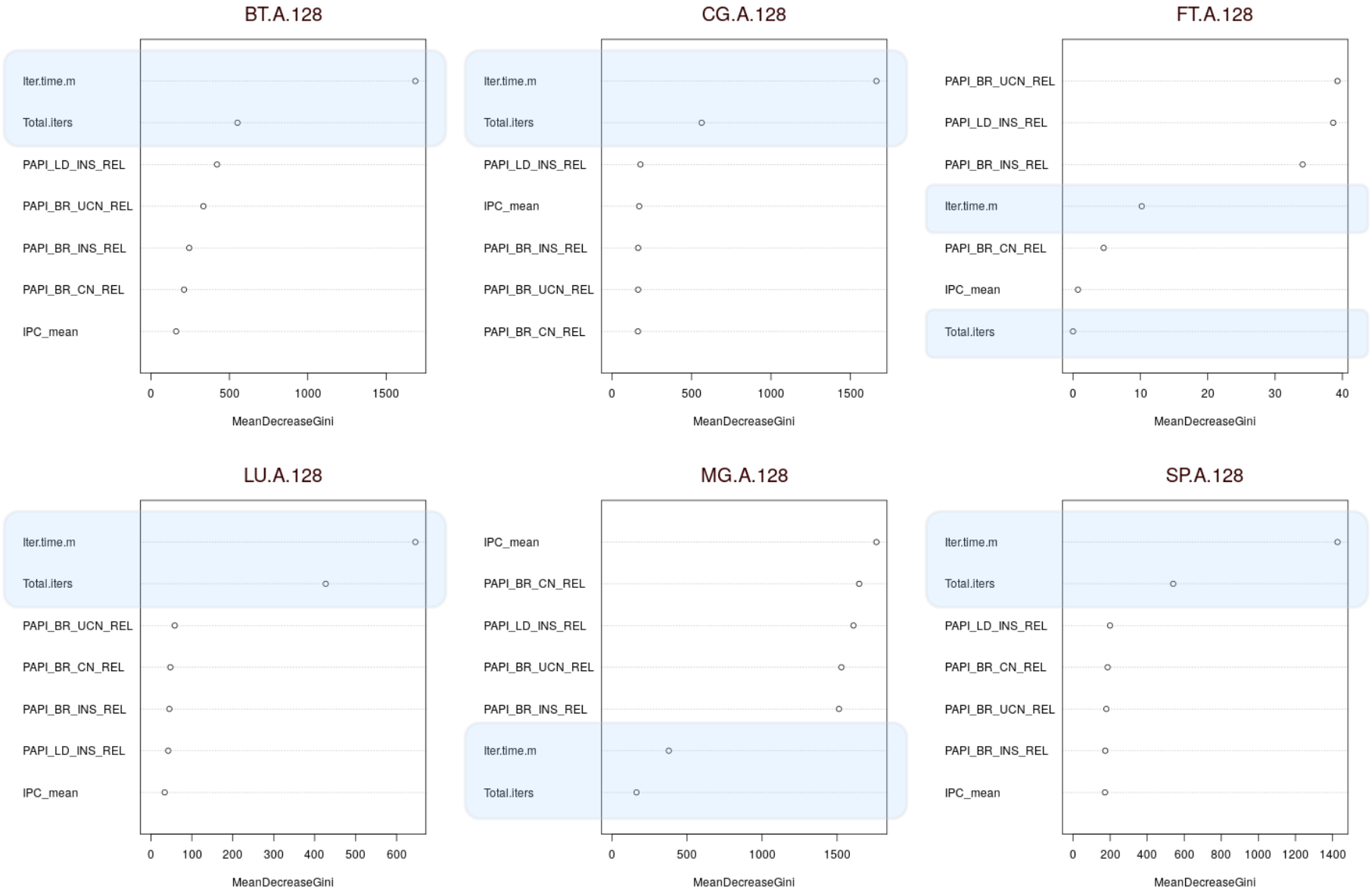






Best features analysis

Analysis of results: **Variable Importance**



Outline for section 8

Results

Lulesh 2.0 (i)

Execution:

- 6014 code lines
- 128 parallel processes
- $\approx 160MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 6ms$

Results

Lulesh 2.0 (i)

Execution:

- 6014 code lines
- 128 parallel processes
- $\approx 160MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 6ms$

Results:

- 30 iterations loop

```
main()
: FOR 1 TO 30 [id=0.0]
: : TimeIncrement()
```

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
lulesh.cc	1263	: : : : CalcForceForNodes()	-	-	-
lulesh.cc	1137	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	10.27us	-	0.28
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	10.51us	22.52KB	0.55
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	8.67us	22.52KB	0.96
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	8.43us	22.52KB	1.08
lulesh-comm.cc	192	: : : : : : MPI_Irecv(0:6)	8.8us	744.0B	1.08
lulesh-comm.cc	201	: : : : : : MPI_Irecv(0:30)	8.54us	744.0B	1.25
lulesh-comm.cc	210	: : : : : : MPI_Irecv(0:26)	8.29us	744.0B	1.25
lulesh-comm.cc	345	: : : : : : MPI_Irecv(0:31)	8.32us	24.0B	0.9
lulesh.cc	1158	: : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**85.2ms**	-	**2.48**
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	20.08us	-	1.59
lulesh-comm.cc	436	: : : : : : MPI_Isend(0:25)	14.95us	22.52KB	1.9
lulesh-comm.cc	477	: : : : : : MPI_Isend(0:5)	10.07us	22.52KB	1.65
lulesh-comm.cc	518	: : : : : : MPI_Isend(0:1)	10.5us	22.52KB	0.93
lulesh-comm.cc	589	: : : : : : MPI_Isend(0:6)	14.79us	744.0B	1.68
lulesh-comm.cc	606	: : : : : : MPI_Isend(0:30)	12.45us	744.0B	1.86
lulesh-comm.cc	623	: : : : : : MPI_Isend(0:26)	13.13us	744.0B	1.66
lulesh-comm.cc	837	: : : : : : MPI_Isend(0:31)	9.48us	24.0B	0.84
lulesh-comm.cc	843	: : : : : : MPI_Waitall(0:)	541.3us	-	0.43
lulesh.cc	1161	: : : : CommSBN()	-	-	-
lulesh-comm.cc	889	: : : : : : MPI_Comm_rank(0:)	9.32us	-	0.49
lulesh-comm.cc	911	: : : : : : MPI_Wait(0:)	9.25us	-	0.6
lulesh-comm.cc	945	: : : : : : MPI_Wait(0:)	8.6us	-	2.48
lulesh-comm.cc	980	: : : : : : MPI_Wait(0:)	8.52us	-	2.35
lulesh-comm.cc	1039	: : : : : : MPI_Wait(0:)	8.41us	-	1.72
lulesh-comm.cc	1053	: : : : : : MPI_Wait(0:)	8.19us	-	1.64
lulesh-comm.cc	1067	: : : : : : MPI_Wait(0:)	8.31us	-	1.98
lulesh-comm.cc	1251	: : : : : : MPI_Wait(0:)	58.99us	-	1.44
lulesh.cc	1267	: : : : CommRecv()	-	-	-
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	7.77us	-	0.73
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	9.03us	45.05KB	0.79
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	7.37us	45.05KB	1.02
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	7.34us	45.05KB	0.95
lulesh-comm.cc	192	: : : : : : MPI_Irecv(0:6)	7.6us	1.45KB	1.12
lulesh-comm.cc	201	: : : : : : MPI_Irecv(0:30)	7.51us	1.45KB	1.33
lulesh-comm.cc	210	: : : : : : MPI_Irecv(0:26)	7.42us	1.45KB	1.13
lulesh-comm.cc	345	: : : : : : MPI_Irecv(0:31)	7.5us	48.0B	0.95
lulesh.cc	1289	: : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**6.15ms**	-	**2.0**
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	14.58us	-	1.63
lulesh-comm.cc	843	: : : : : : MPI_Waitall(0:)	10.26us	-	0.35
lulesh.cc	1292	: : : : CommSyncPosVel()	-	-	-
lulesh-comm.cc	1310	: : : : : : MPI_Comm_rank(0:)	22.14us	-	0.73
lulesh-comm.cc	1332	: : : : : : MPI_Wait(0:)	274.73us	-	0.79
lulesh-comm.cc	1366	: : : : : : MPI_Wait(0:)	72.95us	-	2.46
lulesh-comm.cc	1402	: : : : : : MPI_Wait(0:)	23.24us	-	2.3
lulesh-comm.cc	1461	: : : : : : MPI_Wait(0:)	198.36us	-	1.79
lulesh-comm.cc	1475	: : : : : : MPI_Wait(0:)	7.83us	-	1.83
lulesh-comm.cc	1489	: : : : : : MPI_Wait(0:)	7.75us	-	2.08
lulesh-comm.cc	1674	: : : : : : MPI_Wait(0:)	333.07us	-	1.73
lulesh.cc	2656	: : : LagrangeElements()	-	-	-
lulesh.cc	2461	: : : : CalcQForElems()	-	-	-
lulesh.cc	1992	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	**101**	**: : : : : : [~ computation ~]**	**29.32ms**	-	**2.25**
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	18.76us	-	1.39
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	12.61us	21.09KB	0.43
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	9.23us	21.09KB	1.01
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	8.79us	21.09KB	1.17
lulesh.cc	2010	: : : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : : [~ computation ~]**	**15.35ms**	-	**1.91**
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	15.24us	-	1.52
lulesh-comm.cc	436	: : : : : : MPI_Isend(0:25)	13.14us	21.09KB	1.9
lulesh-comm.cc	477	: : : : : : MPI_Isend(0:5)	10.28us	21.09KB	1.89
lulesh-comm.cc	518	: : : : : : MPI_Isend(0:1)	10.57us	21.09KB	1.26
lulesh-comm.cc	843	: : : : : : MPI_Waitall(0:)	291.68us	-	0.45
lulesh.cc	2014	: : : : : CommMonoQ()	-	-	-
lulesh-comm.cc	1734	: : : : : : MPI_Comm_rank(0:)	9.51us	-	0.45
lulesh-comm.cc	1757	: : : : : : MPI_Wait(0:)	8.88us	-	0.81
lulesh-comm.cc	1791	: : : : : : MPI_Wait(0:)	8.78us	-	2.04
lulesh-comm.cc	1824	: : : : : : MPI_Wait(0:)	8.53us	-	2.13
		: END LOOP	-	-	-

Results

Lulesh 2.0 (i)

Execution:

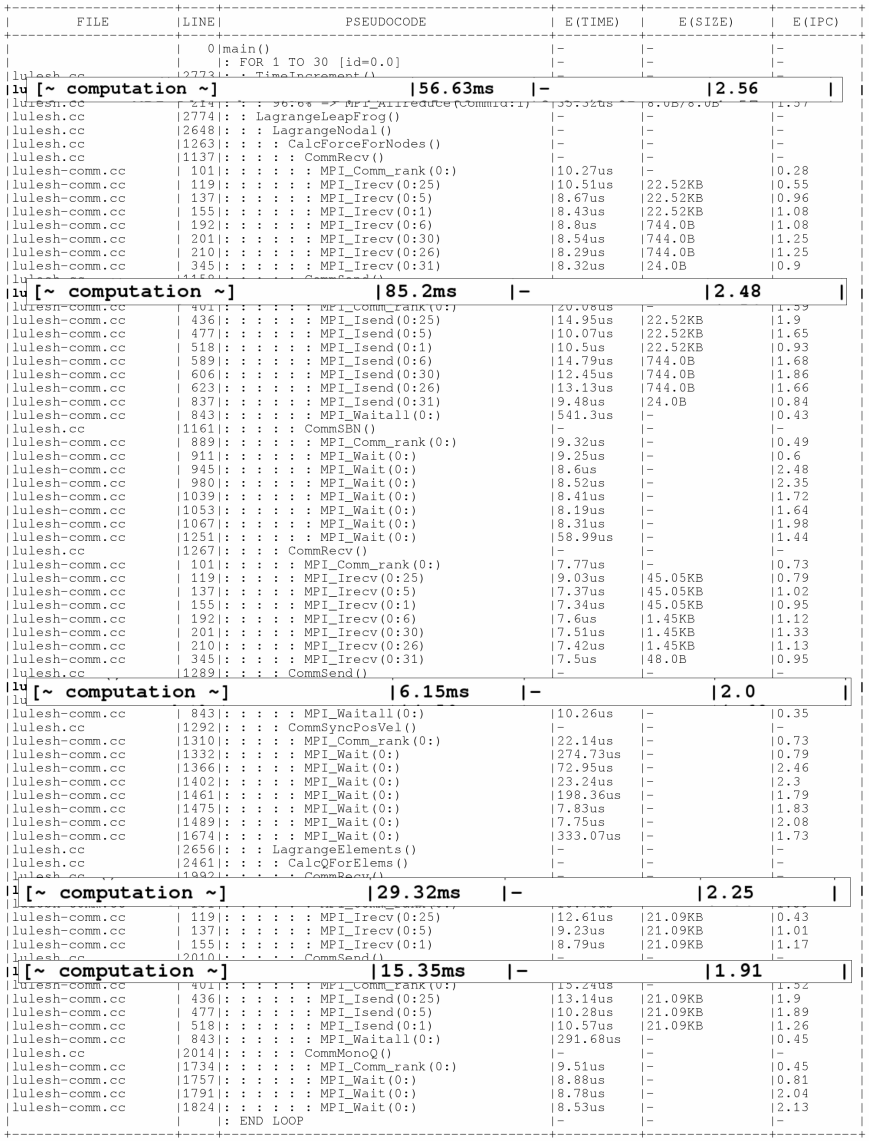
- 6014 code lines
- 128 parallel processes
- $\approx 160MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 6ms$

Results:

- 30 iterations loop
- 5 long CPU burst



Results

Lulesh 2.0 (i)

Execution:

- 6014 code lines
- 128 parallel processes
- $\approx 160MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 6ms$

Results:

- 30 iterations loop
- 5 long CPU burst
- 1 Data conditioned MPI

```
TimeIncrement()
: [~ computation ~]
: 96.6% => MPI_Allreduce(CommId:1)
LagrangeLeapFrog()
```

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
lulesh.cc					-
lulesh.cc					**2.56**
lulesh.cc					1.57
lulesh.cc					-
lulesh.cc					-
lulesh.cc					-
lulesh.cc					-
lulesh-comm.c					0.28
lulesh-comm.cc	119	: : : : : MPI_Irecv(0:25)	10.51us	22.52KB	0.55
lulesh-comm.cc	137	: : : : : MPI_Irecv(0:5)	8.67us	22.52KB	0.96
lulesh-comm.cc	155	: : : : : MPI_Irecv(0:1)	8.43us	22.52KB	1.08
lulesh-comm.cc	192	: : : : : MPI_Irecv(0:6)	8.8us	744.0B	1.08
lulesh-comm.cc	201	: : : : : MPI_Irecv(0:30)	8.54us	744.0B	1.25
lulesh-comm.cc	210	: : : : : MPI_Irecv(0:26)	8.29us	744.0B	1.25
lulesh-comm.cc	345	: : : : : MPI_Irecv(0:31)	8.32us	24.0B	0.9
lulesh.cc	1158	: : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**85.2ms**	**-**	**2.48**
lulesh-comm.cc	401	: : : : : MPI_Comm_rank(0:)	20.08us	-	1.59
lulesh-comm.cc	436	: : : : : MPI_Isend(0:25)	14.95us	22.52KB	1.9
lulesh-comm.cc	477	: : : : : MPI_Isend(0:5)	10.07us	22.52KB	1.65
lulesh-comm.cc	518	: : : : : MPI_Isend(0:1)	10.5us	22.52KB	0.93
lulesh-comm.cc	589	: : : : : MPI_Isend(0:6)	14.79us	744.0B	1.68
lulesh-comm.cc	606	: : : : : MPI_Isend(0:30)	12.45us	744.0B	1.86
lulesh-comm.cc	623	: : : : : MPI_Isend(0:26)	13.13us	744.0B	1.66
lulesh-comm.cc	837	: : : : : MPI_Isend(0:31)	9.48us	24.0B	0.84
lulesh-comm.cc	843	: : : : : MPI_Waitall(0:)	541.3us	-	0.43
lulesh.cc	1161	: : : : CommSBN()	-	-	-
lulesh-comm.cc	889	: : : : : MPI_Comm_rank(0:)	9.32us	-	0.49
lulesh-comm.cc	911	: : : : : MPI_Wait(0:)	9.25us	-	0.6
lulesh-comm.cc	945	: : : : : MPI_Wait(0:)	8.6us	-	2.48
lulesh-comm.cc	980	: : : : : MPI_Wait(0:)	8.52us	-	2.35
lulesh-comm.cc	1039	: : : : : MPI_Wait(0:)	8.41us	-	1.72
lulesh-comm.cc	1053	: : : : : MPI_Wait(0:)	8.19us	-	1.64
lulesh-comm.cc	1067	: : : : : MPI_Wait(0:)	8.31us	-	1.98
lulesh-comm.cc	1251	: : : : : MPI_Wait(0:)	58.99us	-	1.44
lulesh.cc	1267	: : : : CommRecv()	-	-	-
lulesh-comm.cc	101	: : : : : MPI_Comm_rank(0:)	7.77us	-	0.73
lulesh-comm.cc	119	: : : : : MPI_Irecv(0:25)	9.03us	45.05KB	0.79
lulesh-comm.cc	137	: : : : : MPI_Irecv(0:5)	7.37us	45.05KB	1.02
lulesh-comm.cc	155	: : : : : MPI_Irecv(0:1)	7.34us	45.05KB	0.95
lulesh-comm.cc	192	: : : : : MPI_Irecv(0:6)	7.6us	1.45KB	1.12
lulesh-comm.cc	201	: : : : : MPI_Irecv(0:30)	7.51us	1.45KB	1.33
lulesh-comm.cc	210	: : : : : MPI_Irecv(0:26)	7.42us	1.45KB	1.13
lulesh-comm.cc	345	: : : : : MPI_Irecv(0:31)	7.5us	48.0B	0.95
lulesh.cc	1289	: : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**6.15ms**	**-**	**2.0**
lulesh-comm.cc	401	: : : : : MPI_Comm_rank(0:)	14.58us	-	1.63
lulesh-comm.cc	843	: : : : : MPI_Waitall(0:)	10.26us	-	0.35
lulesh.cc	1292	: : : : CommSyncPosVel()	-	-	-
lulesh-comm.cc	1310	: : : : : MPI_Comm_rank(0:)	22.14us	-	0.73
lulesh-comm.cc	1332	: : : : : MPI_Wait(0:)	274.73us	-	0.79
lulesh-comm.cc	1366	: : : : : MPI_Wait(0:)	72.95us	-	2.46
lulesh-comm.cc	1402	: : : : : MPI_Wait(0:)	23.24us	-	2.3
lulesh-comm.cc	1461	: : : : : MPI_Wait(0:)	198.36us	-	1.79
lulesh-comm.cc	1475	: : : : : MPI_Wait(0:)	7.83us	-	1.83
lulesh-comm.cc	1489	: : : : : MPI_Wait(0:)	7.75us	-	2.08
lulesh-comm.cc	1674	: : : : : MPI_Wait(0:)	333.07us	-	1.73
lulesh.cc	2656	: : : LagrangeElements()	-	-	-
lulesh.cc	2461	: : : : CalcQForElems()	-	-	-
lulesh.cc	1992	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	**101**	**: : : : : [~ computation ~]**	**29.32ms**	**-**	**2.25**
lulesh-comm.cc	101	: : : : : MPI_Comm_rank(0:)	18.76us	-	1.39
lulesh-comm.cc	119	: : : : : MPI_Irecv(0:25)	12.61us	21.09KB	0.43
lulesh-comm.cc	137	: : : : : MPI_Irecv(0:5)	9.23us	21.09KB	1.01
lulesh-comm.cc	155	: : : : : MPI_Irecv(0:1)	8.79us	21.09KB	1.17
lulesh.cc	2010	: : : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**15.35ms**	**-**	**1.91**
lulesh-comm.cc	401	: : : : : MPI_Comm_rank(0:)	15.24us	-	1.52
lulesh-comm.cc	436	: : : : : MPI_Isend(0:25)	13.14us	21.09KB	1.9
lulesh-comm.cc	477	: : : : : MPI_Isend(0:5)	10.28us	21.09KB	1.89
lulesh-comm.cc	518	: : : : : MPI_Isend(0:1)	10.57us	21.09KB	1.26
lulesh-comm.cc	843	: : : : : MPI_Waitall(0:)	291.68us	-	0.45
lulesh.cc	2014	: : : : CommMonoQ()	-	-	-
lulesh-comm.cc	1734	: : : : : MPI_Comm_rank(0:)	9.51us	-	0.45
lulesh-comm.cc	1757	: : : : : MPI_Wait(0:)	8.88us	-	0.81
lulesh-comm.cc	1791	: : : : : MPI_Wait(0:)	8.78us	-	2.04
lulesh-comm.cc	1824	: : : : : MPI_Wait(0:)	8.53us	-	2.13
		: END LOOP	-	-	-

Results

Lulesh 2.0 (i)

Execution:

- 6014 code lines
- 128 parallel processes
- $\approx 160MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 6ms$

Results:

- 30 iterations loop
- 5 long CPU burst
- 1 Data conditioned MPI
- 56 no conditioned MPI

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 30 [id=0.0]	-	-	-
lulesh.cc	2773	: : TimeIncrement()	-	-	-
lulesh.cc	**214**	**: : : [~ computation ~]**	**56.63ms**	**-**	**2.56**
lulesh.cc	214	: : : 96.6% => MPI_Allreduce(CommId:1)	35.32us	8.0B/8.0B	1.57
lulesh.cc	2774	: : LagrangeLeapFrog()	-	-	-
lulesh.cc	2648	: : : LagrangeNodal()	-	-	-
lulesh.cc	1263	: : : : CalcForceForNodes()	-	-	-
lulesh.cc	1137	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	10.27us	-	0.28
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	10.51us	22.52KB	0.55
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	8.67us	22.52KB	0.96
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	8.43us	22.52KB	1.08
lulesh-comm.cc	192	: : : : : : MPI_Irecv(0:6)	8.8us	744.0B	1.08
lulesh-comm.cc	201	: : : : : : MPI_Irecv(0:30)	8.54us	744.0B	1.25
lulesh-comm.cc	210	: : : : : : MPI_Irecv(0:26)	8.29us	744.0B	1.25
lulesh-comm.cc	345	: : : : : : MPI_Irecv(0:31)	8.32us	24.0B	0.9
lulesh.cc	1158	: : : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : : [~ computation ~]**	**85.2ms**	**-**	**2.48**
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	20.08us	-	1.59
lulesh-comm.cc	436	: : : : : : MPI_Isend(0:25)	14.95us	22.52KB	1.9
lulesh-comm.cc	477	: : : : : : MPI_Isend(0:5)	10.07us	22.52KB	1.65
lulesh-comm.cc	518	: : : : : : MPI_Isend(0:1)	10.5us	22.52KB	0.93
lulesh-comm.cc	589	: : : : : : MPI_Isend(0:6)	14.79us	744.0B	1.68
lulesh-comm.cc	606	: : : : : : MPI_Isend(0:30)	12.45us	744.0B	1.86
lulesh-comm.cc	623	: : : : : : MPI_Isend(0:26)	13.13us	744.0B	1.66
lulesh-comm.cc	837	: : : : : : MPI_Isend(0:31)	9.48us	24.0B	0.84
lulesh-comm.cc	843	: : : : : : MPI_Waitall(0:)	541.3us	-	0.43
lulesh.cc	1161	: : : : : CommSBN()	-	-	-
lulesh-comm.cc	889	: : : : : : MPI_Comm_rank(0:)	9.32us	-	0.49
lulesh-comm.cc	911	: : : : : : MPI_Wait(0:)	9.25us	-	0.6
lulesh-comm.cc	945	: : : : : : MPI_Wait(0:)	8.6us	-	2.48
lulesh-comm.cc	980	: : : : : : MPI_Wait(0:)	8.52us	-	2.35
lulesh-comm.cc	1039	: : : : : : MPI_Wait(0:)	8.41us	-	1.72
lulesh-comm.cc	1053	: : : : : : MPI_Wait(0:)	8.19us	-	1.64
lulesh-comm.cc	1067	: : : : : : MPI_Wait(0:)	8.31us	-	1.98
lulesh-comm.cc	1251	: : : : : : MPI_Wait(0:)	58.99us	-	1.44
lulesh.cc	1267	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	7.77us	-	0.73
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	9.03us	45.05KB	0.79
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	7.37us	45.05KB	1.02
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	7.34us	45.05KB	0.95
lulesh-comm.cc	192	: : : : : : MPI_Irecv(0:6)	7.6us	1.45KB	1.12
lulesh-comm.cc	201	: : : : : : MPI_Irecv(0:30)	7.51us	1.45KB	1.33
lulesh-comm.cc	210	: : : : : : MPI_Irecv(0:26)	7.42us	1.45KB	1.13
lulesh-comm.cc	345	: : : : : : MPI_Irecv(0:31)	7.5us	48.0B	0.95
lulesh.cc	1289	: : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**6.15ms**	**-**	**2.0**
lulesh-comm.cc	401	: : : : : MPI_Comm_rank(0:)	14.58us	-	1.63
lulesh-comm.cc	843	: : : : : MPI_Waitall(0:)	10.26us	-	0.35
lulesh.cc	1292	: : : : CommSyncPosVel()	-	-	-
lulesh-comm.cc	1310	: : : : : MPI_Comm_rank(0:)	22.14us	-	0.73
lulesh-comm.cc	1332	: : : : : MPI_Wait(0:)	274.73us	-	0.79
lulesh-comm.cc	1366	: : : : : MPI_Wait(0:)	72.95us	-	2.46
lulesh-comm.cc	1402	: : : : : MPI_Wait(0:)	23.24us	-	2.3
lulesh-comm.cc	1461	: : : : : MPI_Wait(0:)	198.36us	-	1.79
lulesh-comm.cc	1475	: : : : : MPI_Wait(0:)	7.83us	-	1.83
lulesh-comm.cc	1489	: : : : : MPI_Wait(0:)	7.75us	-	2.08
lulesh-comm.cc	1674	: : : : : MPI_Wait(0:)	333.07us	-	1.73
lulesh.cc	2656	: : : LagrangeElements()	-	-	-
lulesh.cc	2461	: : : : CalcQForElems()	-	-	-
lulesh.cc	1992	: : : : : CommRecv()	-	-	-
lulesh-comm.cc	**101**	**: : : : : : [~ computation ~]**	**29.32ms**	**-**	**2.25**
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	18.76us	-	1.39
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	12.61us	21.09KB	0.43
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	9.23us	21.09KB	1.01
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	8.79us	21.09KB	1.17
lulesh.cc	2010	: : : : : CommSend()	-	-	-
lulesh-comm.cc	**401**	**: : : : : : [~ computation ~]**	**15.35ms**	**-**	**1.91**
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	15.24us	-	1.52
lulesh-comm.cc	436	: : : : : : MPI_Isend(0:25)	13.14us	21.09KB	1.9
lulesh-comm.cc	477	: : : : : : MPI_Isend(0:5)	10.28us	21.09KB	1.89
lulesh-comm.cc	518	: : : : : : MPI_Isend(0:1)	10.57us	21.09KB	1.26
lulesh-comm.cc	843	: : : : : : MPI_Waitall(0:)	291.68us	-	0.45
lulesh.cc	2014	: : : : : CommMonoQ()	-	-	-
lulesh-comm.cc	1734	: : : : : : MPI_Comm_rank(0:)	9.51us	-	0.45
lulesh-comm.cc	1757	: : : : : : MPI_Wait(0:)	8.88us	-	0.81
lulesh-comm.cc	1791	: : : : : : MPI_Wait(0:)	8.78us	-	2.04
lulesh-comm.cc	1824	: : : : : : MPI_Wait(0:)	8.53us	-	2.13
		: END LOOP	-	-	-

Results

Lulesh 2.0 (ii)

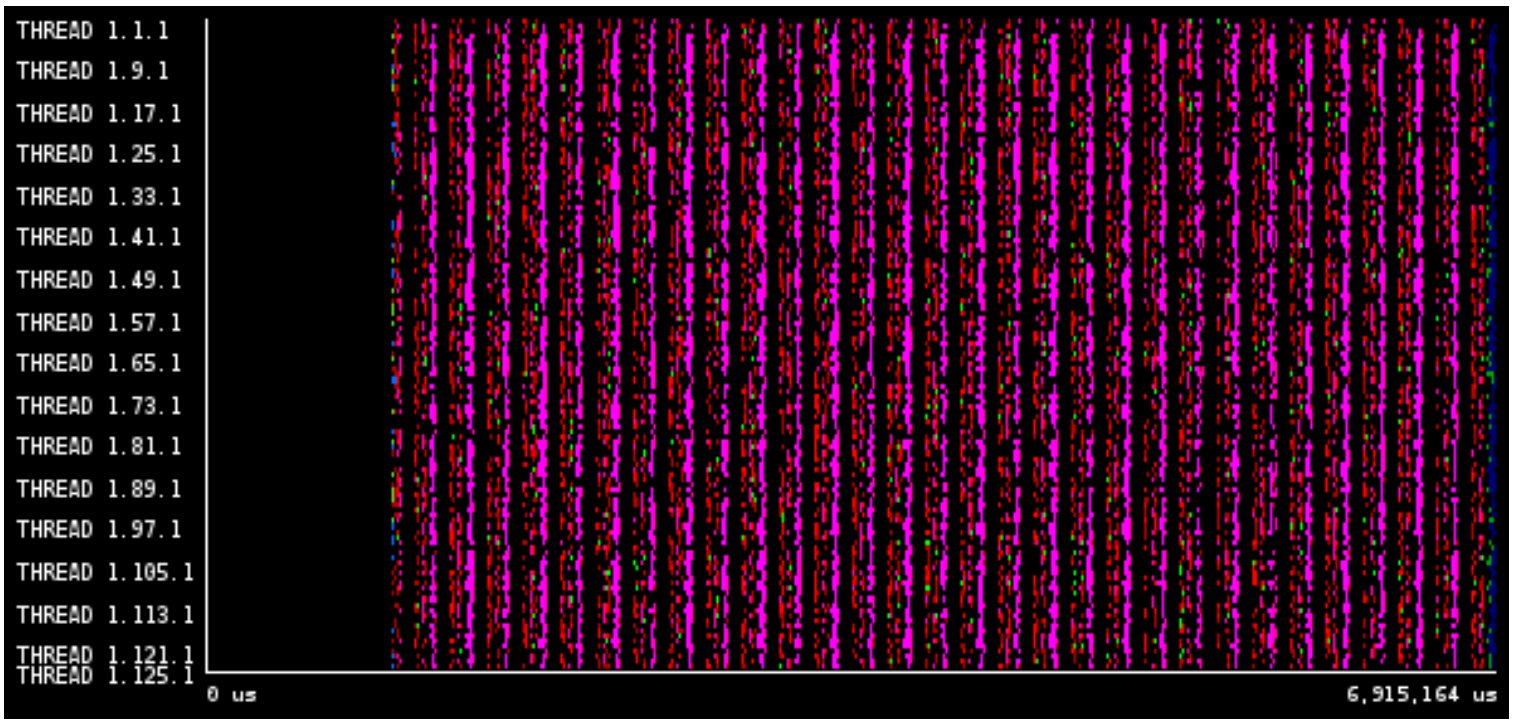


Figure: Lulesh 2.0 128 MPI ranks – 30 iterations

Results

Lulesh 2.0 (ii)

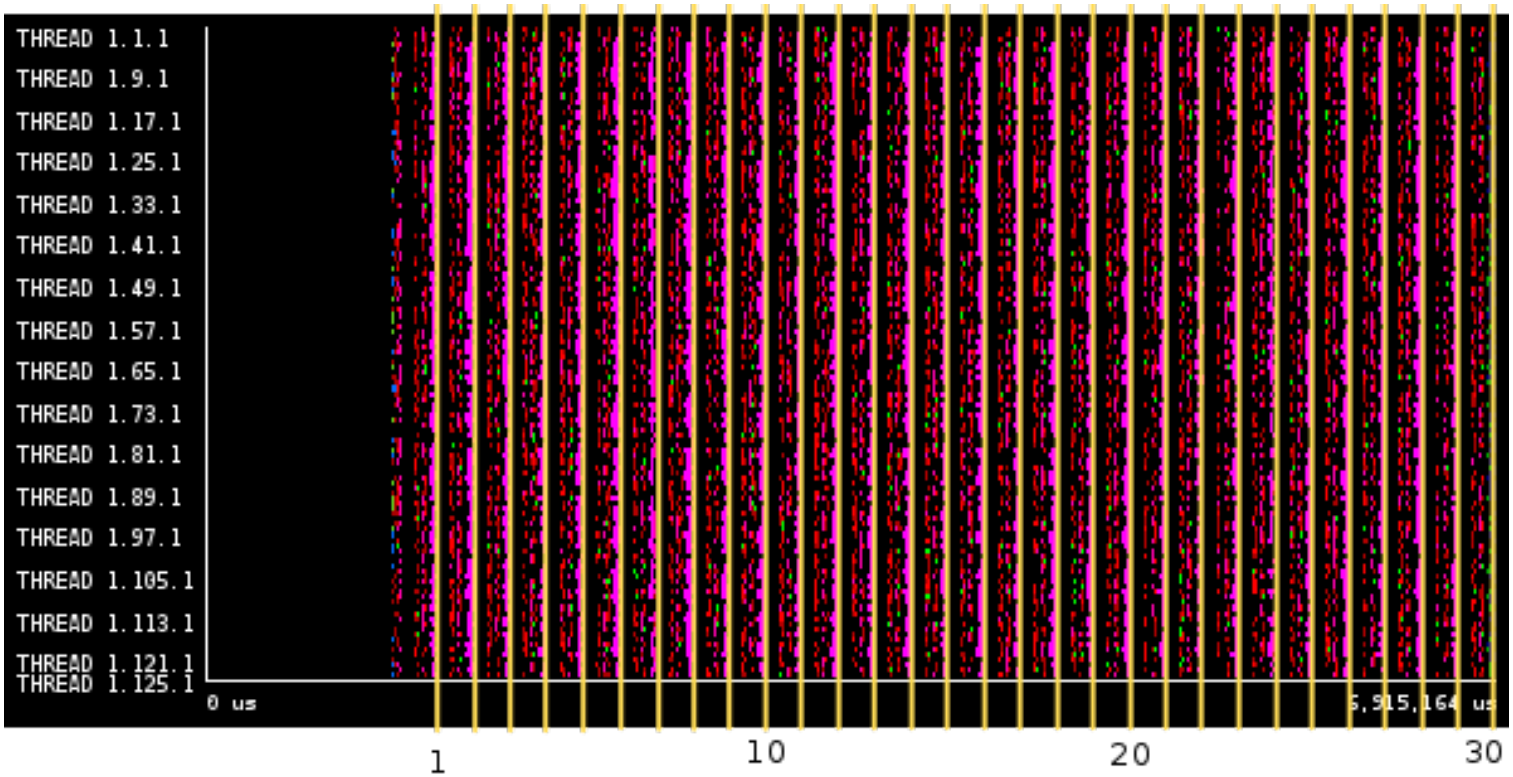
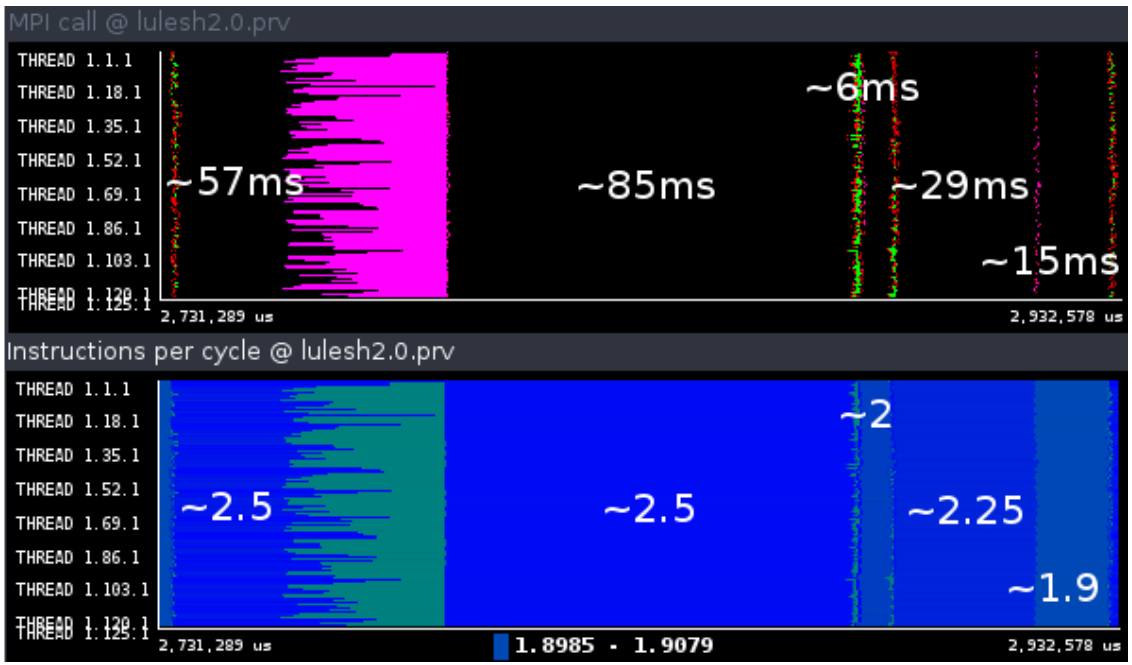


Figure: Lulesh 2.0 128 MPI ranks – 30 iterations

Results

Lulesh 2.0 (iii)

		: FOR 1 TO 30 [id=0.0]	-	-	-	
lulesh.cc	2773	: : TimeIncrement()	-	-	-	
lulesh.cc	**214**	**: : : [~ computation ~]**	**56.63ms**	**-**	**2.56**	
lulesh.cc	214	: : : 96.6% => MPI_Allreduce(CommId:1)	35.32us	8.0B/8.0B	1.57	
lulesh.cc	2774	: : LagrangeLeapFrog()	-	-	-	
		...				
lulesh.cc	1158	: : : : : CommSend()	-	-	-	
lulesh-comm.cc	**401**	**: : : : : : [~ computation ~]**	**85.2ms**	**-**	**2.48**	
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	20.08us	-	1.59	
		...				
lulesh-comm.cc	345	: : : : : MPI_Irecv(0:31)	7.5us	48.0B	0.95	
lulesh.cc	1289	: : : : CommSend()	-	-	-	
lulesh-comm.cc	**401**	**: : : : : [~ computation ~]**	**6.15ms**	**-**	**2.0**	
lulesh-comm.cc	401	: : : : : MPI_Comm_rank(0:)	14.58us	-	1.63	
lulesh-comm.cc	843	: : : : : MPI_Waitall(0:)	10.26us	-	0.35	
		...				
lulesh.cc	1992	: : : : : CommRecv()	-	-	-	
lulesh-comm.cc	**101**	**: : : : : : [~ computation ~]**	**29.32ms**	**-**	**2.25**	
lulesh-comm.cc	101	: : : : : : MPI_Comm_rank(0:)	18.76us	-	1.39	
lulesh-comm.cc	119	: : : : : : MPI_Irecv(0:25)	12.61us	21.09KB	0.43	
lulesh-comm.cc	137	: : : : : : MPI_Irecv(0:5)	9.23us	21.09KB	1.01	
lulesh-comm.cc	155	: : : : : : MPI_Irecv(0:1)	8.79us	21.09KB	1.17	
lulesh.cc	2010	: : : : : CommSend()	-	-	-	
lulesh-comm.cc	**401**	**: : : : : : [~ computation ~]**	**15.35ms**	**-**	**1.91**	
lulesh-comm.cc	401	: : : : : : MPI_Comm_rank(0:)	15.24us	-	1.52	



Results

CG Class A (i)

Execution:

- $\approx$ 2000 code lines
- 32 parallel processes
- $\approx 150MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 100us$

Results

CG Class A (i)

Execution:

- $\approx$ 2000 code lines
- 32 parallel processes
- $\approx 150MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 100us$

Results:

- 15 iterations loop

```
+-----+----+-----------------------------+----------+--------------+----------+
|      |    0|MAIN__()                    |-         |-             |-         |
|      |     |: FOR 1 TO 15               |-         |-             |-         |
|cg.f  |  475|: : conj_grad_()            |-         |-             |-         |
|      |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f  |1089|: : : : MPI_Irecv(0:1,2,4)   |7.23us    |8.0B          |1.76      |
|cg.f  |1096|: : : : MPI_Send(0:1,2,4)    |6.81us    |8.0B          |1.34      |
|cg.f  |1097|: : : : MPI_Wait(0:)         |46.79us   |-             |1.32      |
|      |     |: : : END LOOP              |-         |-             |-         |
|      |     |: : : FOR 1 TO 25.0         |-         |-             |-         |
|      |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f  |1137|: : : : : [~ computation ~]  |115.51us  |-             |2.32      |
|cg.f  |1137|: : : : : MPI_Irecv(0:1,2,4) |7.31us    |13.67KB       |1.27      |
|cg.f  |1144|: : : : : MPI_Send(0:1,2,4)  |10.52us   |13.67KB       |1.42      |
|cg.f  |1145|: : : : : MPI_Wait(0:)       |12.07us   |-             |1.3       |
|      |     |: : : : END LOOP            |-         |-             |-         |
|cg.f  |1163|: : : : MPI_Irecv(0:0)       |6.7us     |13.67KB       |2.48      |
|cg.f  |1171|: : : : MPI_Send(0:0)        |7.04us    |13.67KB       |1.45      |
|cg.f  |1172|: : : : MPI_Wait(0:)         |6.45us    |-             |1.28      |
|      |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f  |1207|: : : : : MPI_Irecv(0:1,2,4) |6.55us    |8.0B          |1.83      |
|cg.f  |1214|: : : : : MPI_Send(0:1,2,4)  |6.98us    |8.0B          |1.52      |
|cg.f  |1216|: : : : : MPI_Wait(0:)       |13.15us   |-             |1.4       |
|      |     |: : : : END LOOP            |-         |-             |-         |
|      |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f  |1262|: : : : : MPI_Irecv(0:1,2,4) |          |8.0B          |2.75      |
|cg.f  |1269|: : : : : MPI_Send(0:1,2,4)  |   15x    |8.0B          |1.51      |
|cg.f  |1270|: : : : : MPI_Wait(0:)       |          |-             |1.39      |
|      |     |: : : : END LOOP            |-         |-             |-         |
|      |     |: : : END LOOP              |-         |-             |-         |
|      |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f  |1320|: : : : [~ computation ~]    |115.65us  |-             |2.31      |
|cg.f  |1320|: : : : MPI_Irecv(0:1,2,4)   |6.94us    |13.67KB       |1.13      |
|cg.f  |1327|: : : : MPI_Send(0:1,2,4)    |10.55us   |13.67KB       |1.43      |
|cg.f  |1328|: : : : MPI_Wait(0:)         |9.14us    |-             |1.25      |
|      |     |: : : END LOOP              |-         |-             |-         |
|cg.f  |1347|: : : MPI_Irecv(0:0)         |6.76us    |13.67KB       |2.47      |
|cg.f  |1355|: : : MPI_Send(0:0)          |7.13us    |13.67KB       |1.44      |
|cg.f  |1356|: : : MPI_Wait(0:)           |6.59us    |-             |1.19      |
|      |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f  |1384|: : : : MPI_Irecv(0:1,2,4)   |6.57us    |8.0B          |2.37      |
|cg.f  |1391|: : : : MPI_Send(0:1,2,4)    |6.63us    |8.0B          |1.51      |
|cg.f  |1392|: : : : MPI_Wait(0:)         |25.38us   |-             |1.38      |
|      |     |: : : END LOOP              |-         |-             |-         |
|      |     |: : FOR 1 TO 3.0            |-         |-             |-         |
|cg.f  | 499|: : : MPI_Irecv(0:1,2,4)     |6.03us    |16.0B         |3.09      |
|cg.f  | 506|: : : MPI_Send(0:1,2,4)      |6.1us     |16.0B         |1.53      |
|cg.f  | 507|: : : MPI_Wait(0:)           |6.48us    |-             |1.35      |
|      |     |: : END LOOP                |-         |-             |-         |
|      |     |: END LOOP                  |-         |-             |-         |
+-----+----+-----------------------------+----------+--------------+----------+
```

Results

CG Class A (i)

Execution:

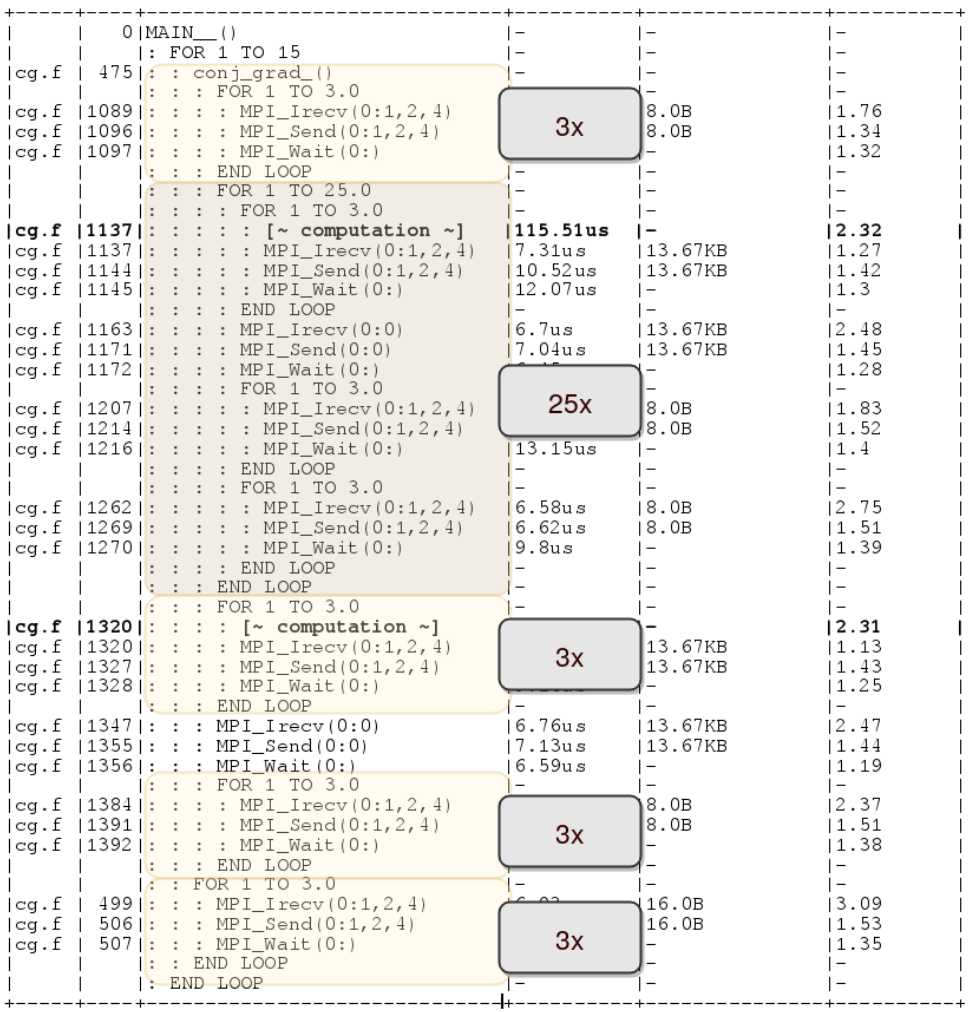
- $\approx$ 2000 code lines
- 32 parallel processes
- $\approx 150MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 100us$

Results:

- 15 iterations loop
- 25 iterations subloop



Results

CG Class A (i)

Execution:

- $\approx$ 2000 code lines
- 32 parallel processes
- $\approx 150MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 100us$

Results:

- 15 iterations loop
- 25 iterations subloop
- Other 3 iterations subloops

```
+-----+----+-----------------------------+----------+--------------+----------+
|     |    0|MAIN__()                     |-         |-             |-         |
|     |     |: FOR 1 TO 15                |-         |-             |-         |
|cg.f | 475|: : conj_grad_()              |-         |-             |-         |
|     |     |: : : FOR 1 TO 3.0           |-         |-             |-         |
|cg.f |1089|: : : : MPI_Irecv(0:1,2,4)    |7.23us    |8.0B          |1.76      |
|cg.f |1096|: : : : MPI_Send(0:1,2,4)     |6.81us    |8.0B          |1.34      |
|cg.f |1097|: : : : MPI_Wait(0:)          |46.79us   |-             |1.32      |
|     |     |: : : END LOOP               |-         |-             |-         |
|     |     |: : : FOR 1 TO 25.0          |-         |-             |-         |
|     |     |: : : : FOR 1 TO 3.0         |-         |-             |-         |
|cg.f |1137|: : : : : [~ computation ~]    |          |-             |2.32      |
|cg.f |1137|: : : : : MPI_Irecv(0:1,2,4)   |   3x     |13.67KB       |1.27      |
|cg.f |1144|: : : : : MPI_Send(0:1,2,4)    |          |13.67KB       |1.42      |
|cg.f |1145|: : : : : MPI_Wait(0:)         |          |-             |1.3       |
|     |     |: : : : END LOOP             |-         |-             |-         |
|cg.f |1163|: : : : MPI_Irecv(0:0)        |6.7us     |13.67KB       |2.48      |
|cg.f |1171|: : : : MPI_Send(0:0)         |7.04us    |13.67KB       |1.45      |
|cg.f |1172|: : : : MPI_Wait(0:)          |6.45us    |-             |1.28      |
|     |     |: : : : FOR 1 TO 3.0         |-         |-             |-         |
|cg.f |1207|: : : : : MPI_Irecv(0:1,2,4)   |   3x     |8.0B          |1.83      |
|cg.f |1214|: : : : : MPI_Send(0:1,2,4)    |          |8.0B          |1.52      |
|cg.f |1216|: : : : : MPI_Wait(0:)         |          |-             |1.4       |
|     |     |: : : : END LOOP             |-         |-             |-         |
|     |     |: : : : FOR 1 TO 3.0         |-         |-             |-         |
|cg.f |1262|: : : : : MPI_Irecv(0:1,2,4)   |   3x     |8.0B          |2.75      |
|cg.f |1269|: : : : : MPI_Send(0:1,2,4)    |          |8.0B          |1.51      |
|cg.f |1270|: : : : : MPI_Wait(0:)         |          |-             |1.39      |
|     |     |: : : : END LOOP             |-         |-             |-         |
|     |     |: : : END LOOP               |-         |-             |-         |
|     |     |: : : FOR 1 TO 3.0           |-         |-             |-         |
|cg.f |1320|: : : : [~ computation ~]      |115.65us  |-             |2.31      |
|cg.f |1320|: : : : MPI_Irecv(0:1,2,4)    |6.94us    |13.67KB       |1.13      |
|cg.f |1327|: : : : MPI_Send(0:1,2,4)     |10.55us   |13.67KB       |1.43      |
|cg.f |1328|: : : : MPI_Wait(0:)          |9.14us    |-             |1.25      |
|     |     |: : : END LOOP               |-         |-             |-         |
|cg.f |1347|: : : MPI_Irecv(0:0)          |6.76us    |13.67KB       |2.47      |
|cg.f |1355|: : : MPI_Send(0:0)           |7.13us    |13.67KB       |1.44      |
|cg.f |1356|: : : MPI_Wait(0:)            |6.59us    |-             |1.19      |
|     |     |: : : FOR 1 TO 3.0           |-         |-             |-         |
|cg.f |1384|: : : : MPI_Irecv(0:1,2,4)    |6.57us    |8.0B          |2.37      |
|cg.f |1391|: : : : MPI_Send(0:1,2,4)     |6.63us    |8.0B          |1.51      |
|cg.f |1392|: : : : MPI_Wait(0:)          |25.38us   |-             |1.38      |
|     |     |: : : END LOOP               |-         |-             |-         |
|     |     |: : FOR 1 TO 3.0             |-         |-             |-         |
|cg.f | 499|: : : MPI_Irecv(0:1,2,4)      |6.03us    |16.0B         |3.09      |
|cg.f | 506|: : : MPI_Send(0:1,2,4)       |6.1us     |16.0B         |1.53      |
|cg.f | 507|: : : MPI_Wait(0:)            |6.48us    |-             |1.35      |
|     |     |: : END LOOP                 |-         |-             |-         |
|     |     |: END LOOP                   |-         |-             |-         |
+-----+----+-----------------------------+----------+--------------+----------+
```

Results

CG Class A (i)

Execution:

- $\approx$ 2000 code lines
- 32 parallel processes
- $\approx 150MB$ tracefile

Analysis:

- Filter all $\delta < 10\%$
- CPU burst $time > 100us$

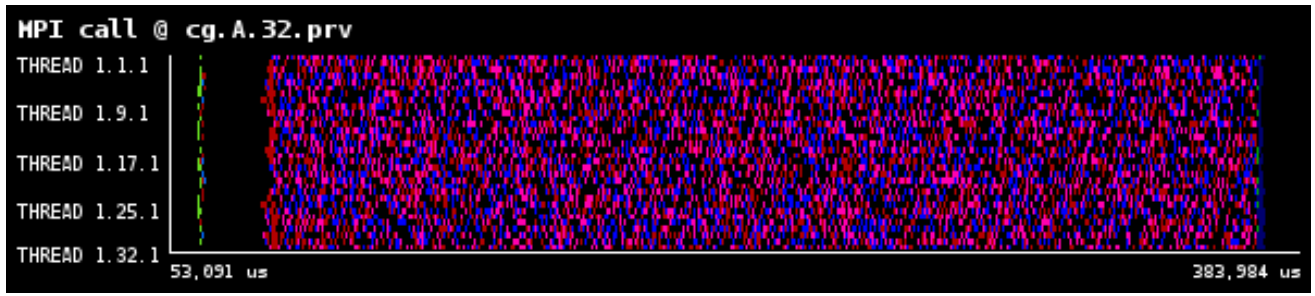
Results:

- 15 iterations loop
- 25 iterations subloop
- Other 3 iterations subloops
- 2 long CPU burst above 100us

```
+-----+----+-----------------------------+----------+--------------+----------+
|     |    0|MAIN__()                    |-         |-             |-         |
|     |     |: FOR 1 TO 15               |-         |-             |-         |
|cg.f | 475|: : conj_grad_()             |-         |-             |-         |
|     |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f |1089|: : : : MPI_Irecv(0:1,2,4)   |7.23us    |8.0B          |1.76      |
|cg.f |1096|: : : : MPI_Send(0:1,2,4)    |6.81us    |8.0B          |1.34      |
|cg.f |1097|: : : : MPI_Wait(0:)         |46.79us   |-             |1.32      |
|     |     |: : : END LOOP              |-         |-             |-         |
|     |     |: : : FOR 1 TO 25.0         |-         |-             |-         |
|     |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f |1137|: : : : : [~ computation ~]   |115.51us  |-             |2.32      |
|cg.f |1137|: : : : : MPI_Irecv(0:1,2,4)  |7.31us    |13.67KB       |1.27      |
|cg.f |1144|: : : : : MPI_Send(0:1,2,4)   |10.52us   |13.67KB       |1.42      |
|cg.f |1145|: : : : : MPI_Wait(0:)        |12.07us   |-             |1.3       |
|     |     |: : : : END LOOP            |-         |-             |-         |
|cg.f |1163|: : : : MPI_Irecv(0:0)        |6.7us     |13.67KB       |2.48      |
|cg.f |1171|: : : : MPI_Send(0:0)         |7.04us    |13.67KB       |1.45      |
|cg.f |1172|: : : : MPI_Wait(0:)          |6.45us    |-             |1.28      |
|     |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f |1207|: : : : : MPI_Irecv(0:1,2,4)  |6.55us    |8.0B          |1.83      |
|cg.f |1214|: : : : : MPI_Send(0:1,2,4)   |6.98us    |8.0B          |1.52      |
|cg.f |1216|: : : : : MPI_Wait(0:)        |13.15us   |-             |1.4       |
|     |     |: : : : END LOOP            |-         |-             |-         |
|     |     |: : : : FOR 1 TO 3.0        |-         |-             |-         |
|cg.f |1262|: : : : : MPI_Irecv(0:1,2,4)  |6.58us    |8.0B          |2.75      |
|cg.f |1269|: : : : : MPI_Send(0:1,2,4)   |6.62us    |8.0B          |1.51      |
|cg.f |1270|: : : : : MPI_Wait(0:)        |9.8us     |-             |1.39      |
|     |     |: : : : END LOOP            |-         |-             |-         |
|     |     |: : : END LOOP              |-         |-             |-         |
|     |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f |1320|: : : : [~ computation ~]     |115.65us  |-             |2.31      |
|cg.f |1320|: : : : MPI_Irecv(0:1,2,4)    |6.94us    |13.67KB       |1.13      |
|cg.f |1327|: : : : MPI_Send(0:1,2,4)     |10.55us   |13.67KB       |1.43      |
|cg.f |1328|: : : : MPI_Wait(0:)          |9.14us    |-             |1.25      |
|     |     |: : : END LOOP              |-         |-             |-         |
|cg.f |1347|: : : MPI_Irecv(0:0)          |6.76us    |13.67KB       |2.47      |
|cg.f |1355|: : : MPI_Send(0:0)           |7.13us    |13.67KB       |1.44      |
|cg.f |1356|: : : MPI_Wait(0:)            |6.59us    |-             |1.19      |
|     |     |: : : FOR 1 TO 3.0          |-         |-             |-         |
|cg.f |1384|: : : : MPI_Irecv(0:1,2,4)    |6.57us    |8.0B          |2.37      |
|cg.f |1391|: : : : MPI_Send(0:1,2,4)     |6.63us    |8.0B          |1.51      |
|cg.f |1392|: : : : MPI_Wait(0:)          |25.38us   |-             |1.38      |
|     |     |: : : END LOOP              |-         |-             |-         |
|     |     |: : FOR 1 TO 3.0            |-         |-             |-         |
|cg.f | 499|: : : MPI_Irecv(0:1,2,4)      |6.03us    |16.0B         |3.09      |
|cg.f | 506|: : : MPI_Send(0:1,2,4)       |6.1us     |16.0B         |1.53      |
|cg.f | 507|: : : MPI_Wait(0:)            |6.48us    |-             |1.35      |
|     |     |: : END LOOP                |-         |-             |-         |
|     |     |: END LOOP                  |-         |-             |-         |
+-----+----+-----------------------------+----------+--------------+----------+
```

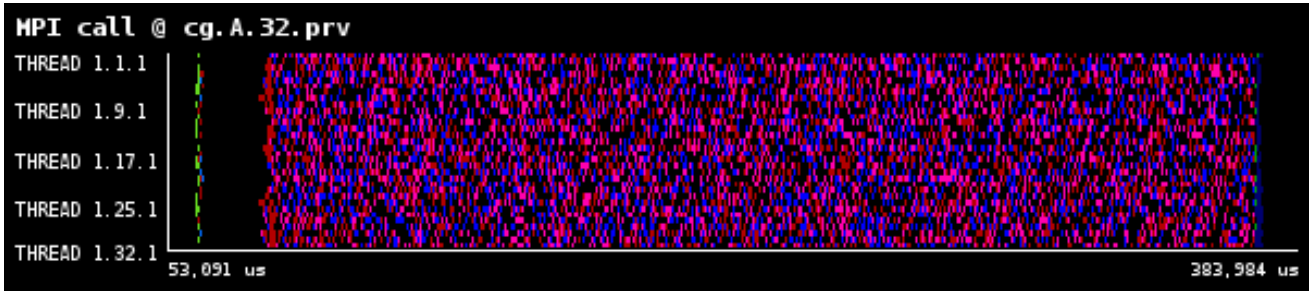
Results

CG Class A(ii)

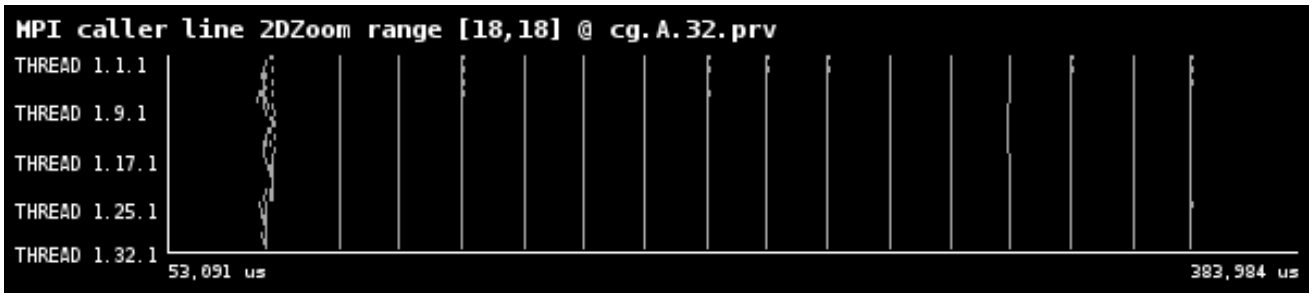


Results

CG Class A(ii)

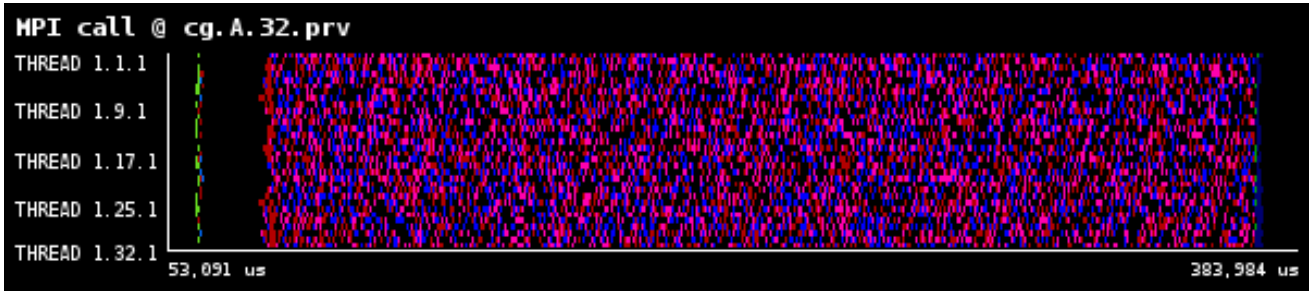


Showing just **MPI_Irecv** on **cg.f : 1089**



Results

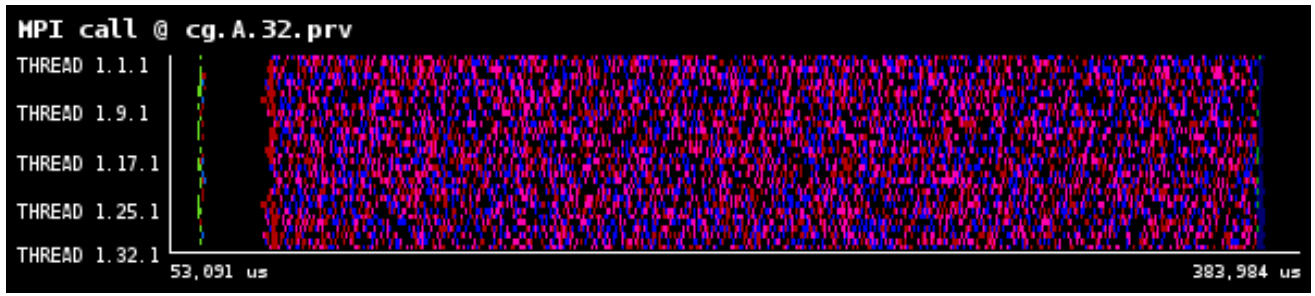
CG Class A(ii)



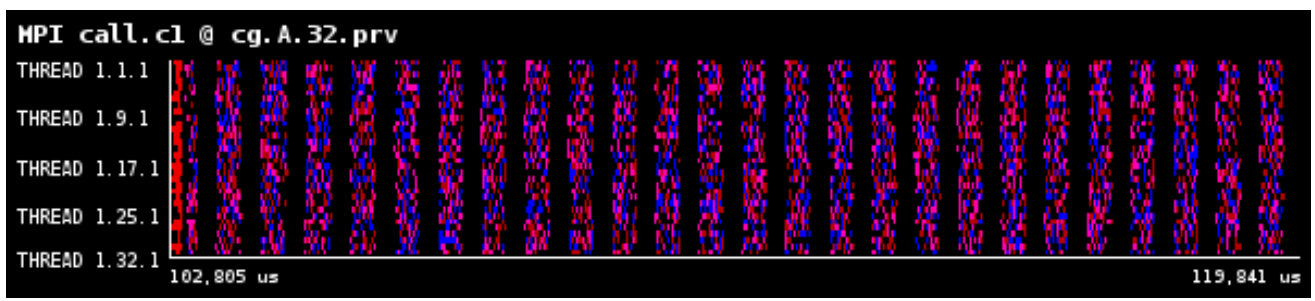
Showing just **MPI_Irecv** on **cg.f : 1089**

Results

CG Class A(ii)

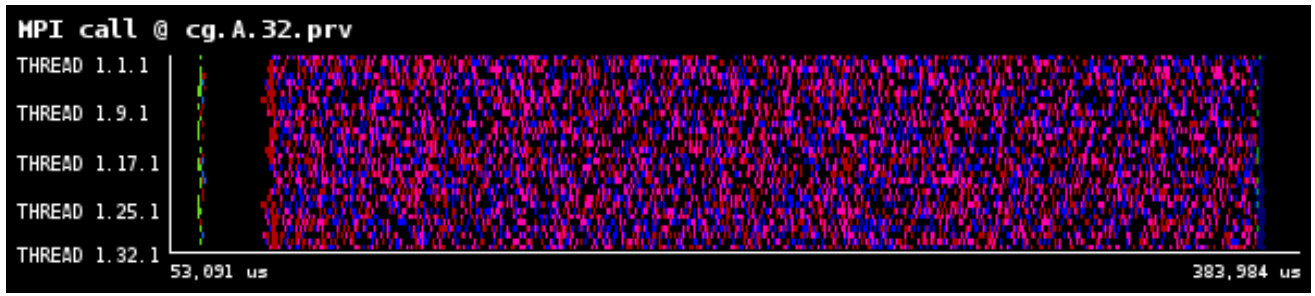


Showing just **MPI_Irecv** on **cg.f : 1089**

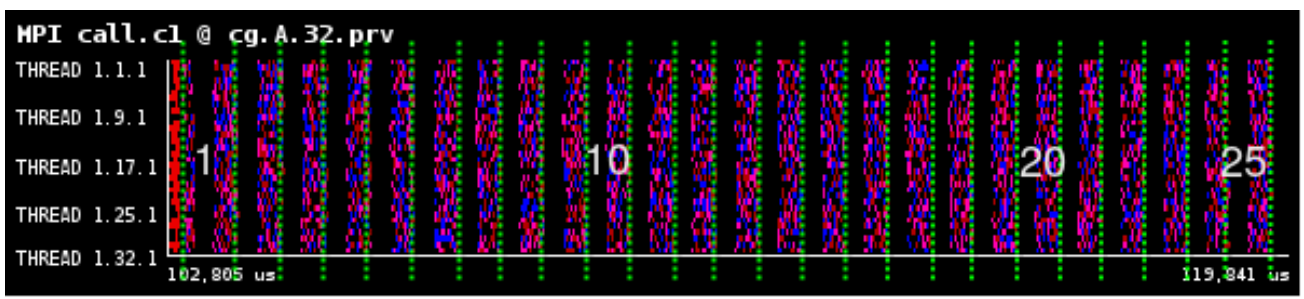


Results

CG Class A(ii)

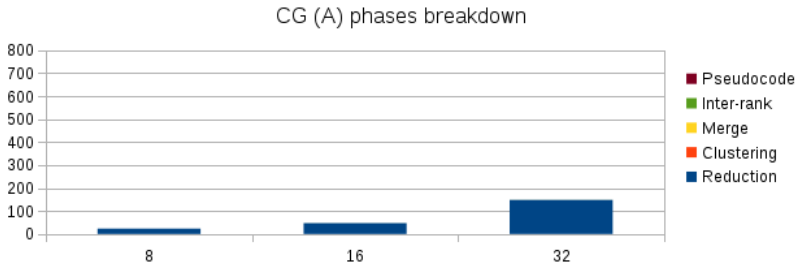


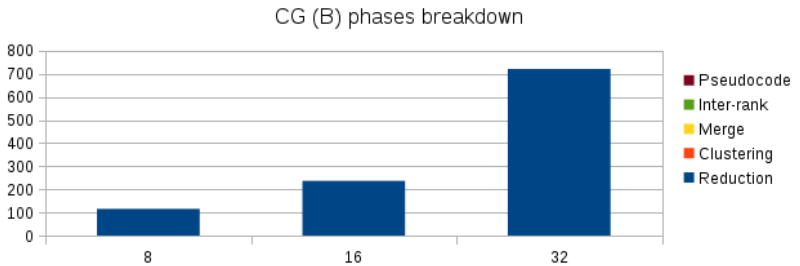
Showing just **MPI_Irecv** on **cg.f : 1089**

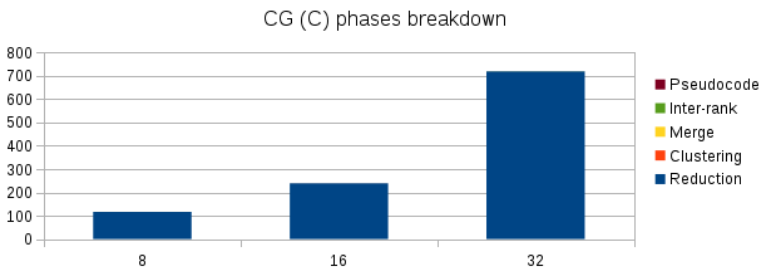


Outline for section 9

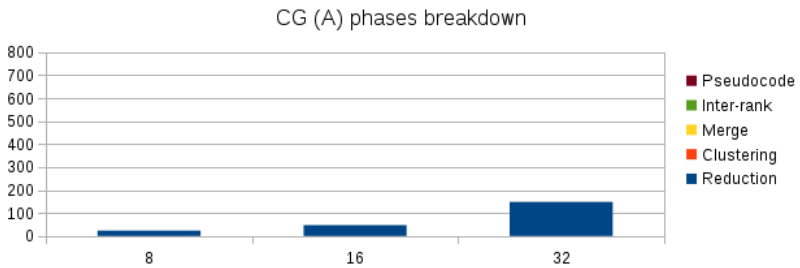
Scalability

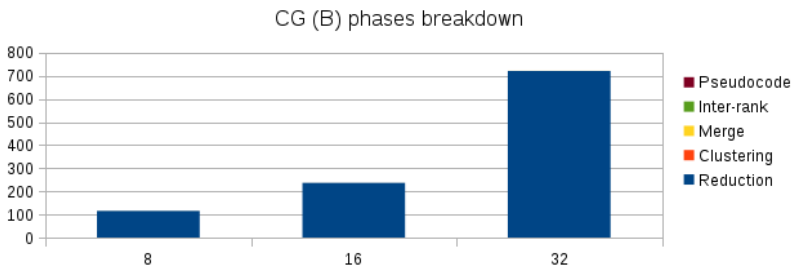


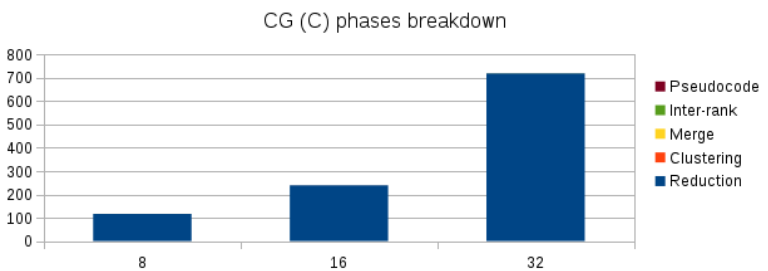




Scalability

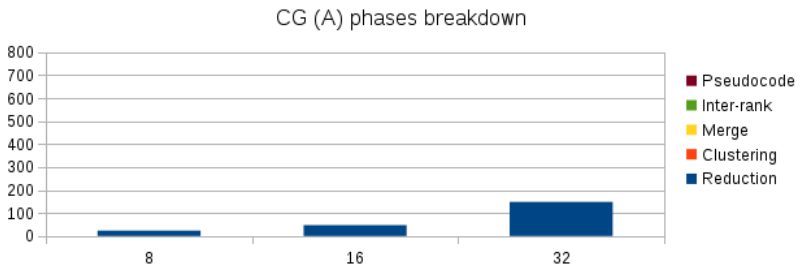


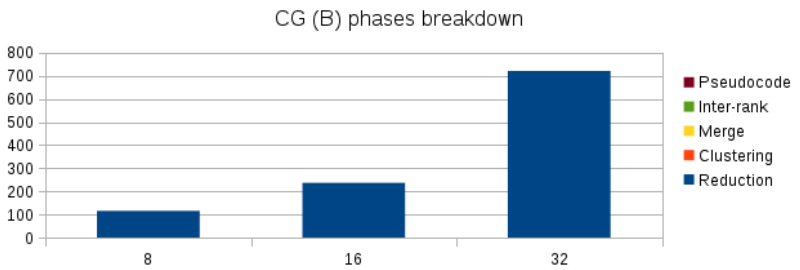


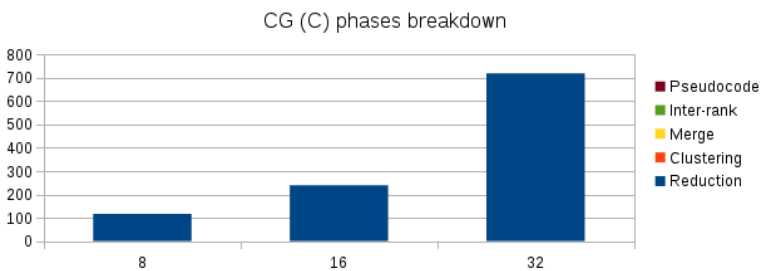


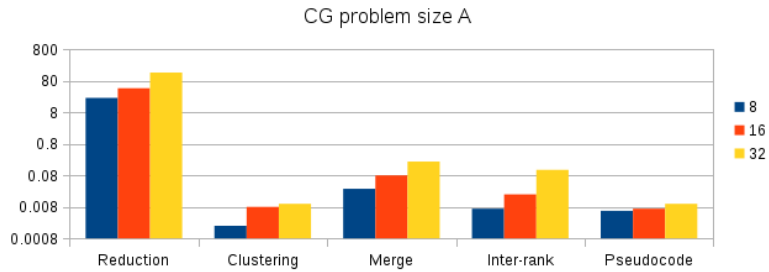
Reduction step is clearly dominating the execution time.

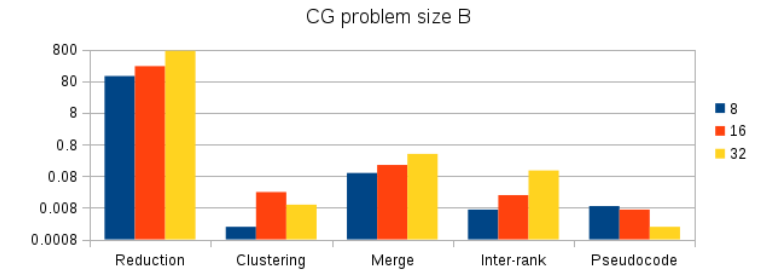
Scalability

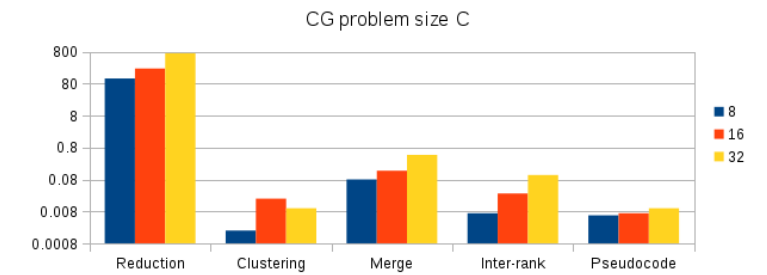






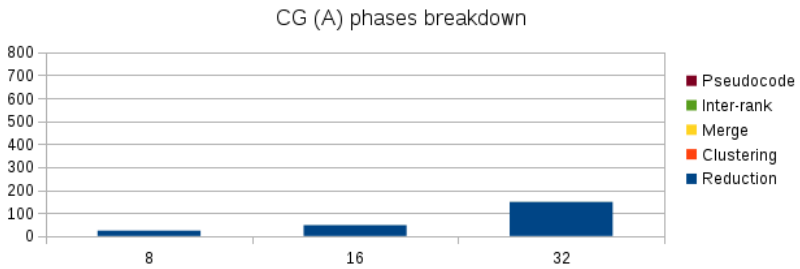


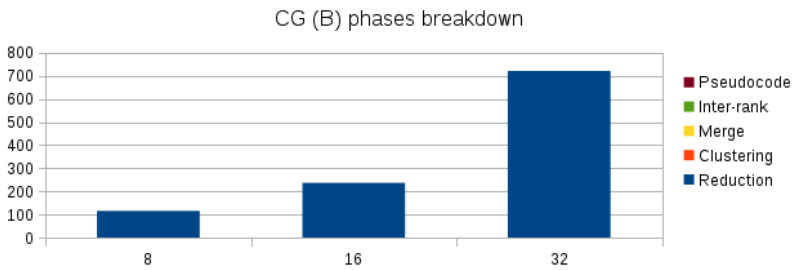


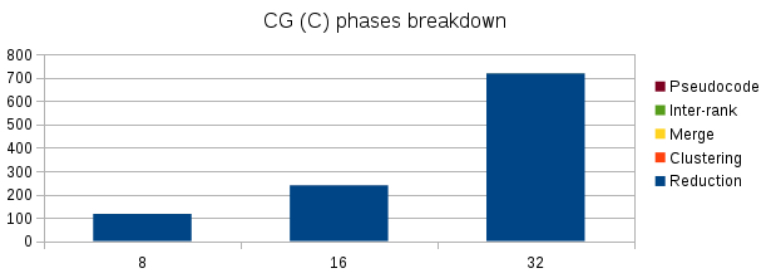


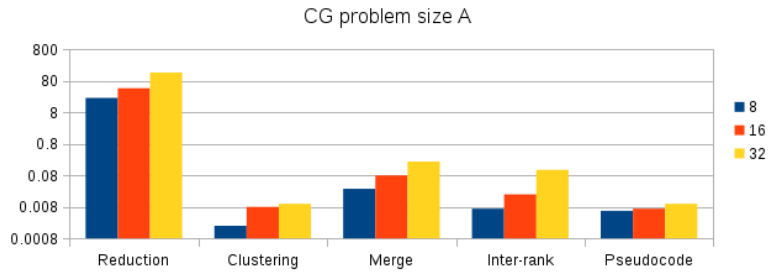
Reduction step is clearly dominating the execution time.

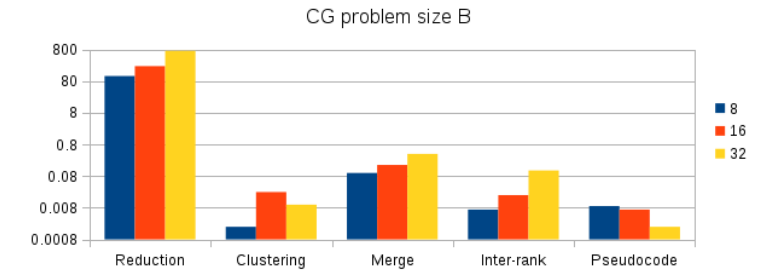
Scalability

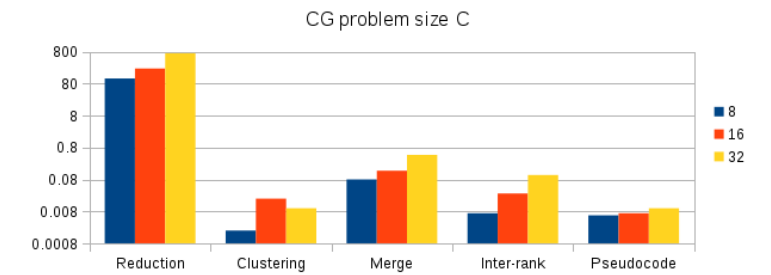








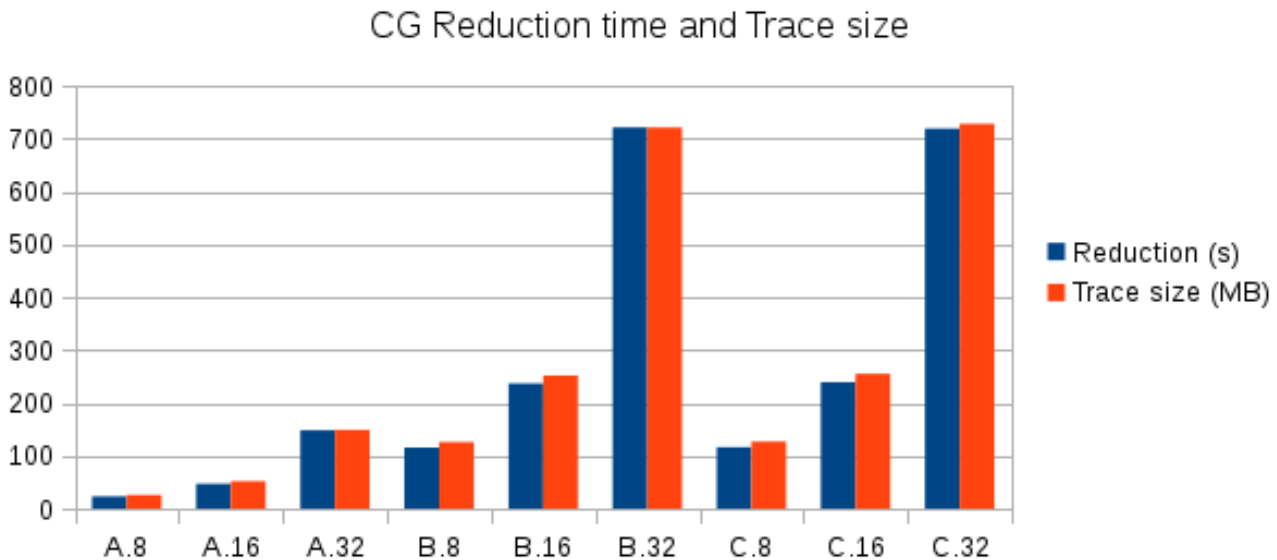




Reduction step is clearly dominating the execution time.

The rest maintains its time even if increasing problem size.

Scalability



Keynote

Reduction time highly correlated with trace size

Outline for section 10

Conclusions

- New approach has been explored.

Conclusions

- New approach has been explored.
- That demonstrates that works well with HPC applications.

Conclusions

- New approach has been explored.
- That demonstrates that works well with HPC applications.
- Iterations mean time and number of iterations demonstrates to be valuable features for classification

Conclusions

- New approach has been explored.
- That demonstrates that works well with HPC applications.
- Iterations mean time and number of iterations demonstrates to be valuable features for classification
- Execution time clearly dominated by Reduction step that presents linear complexity with trace size.

Outline for section 11

Future work

- Improve memory usage by compress timestamps.

Future work

- Improve memory usage by compress timestamps.
- More research for select the best features set that avoids cluster aliasing/split.

Future work

- Improve memory usage by compress timestamps.
- More research for select the best features set that avoids cluster aliasing/split.
- Improve Reduce phase times by, e.g. Parallelize the reduction using for example well-known infrastructures like Hadoop.

Future work

- Improve memory usage by compress timestamps.
- More research for select the best features set that avoids cluster aliasing/split.
- Improve Reduce phase times by, e.g. Parallelize the reduction using for example well-known infrastructures like Hadoop.
- Exploring the possibility to use sampling techniques to get more detailed view of the structure of an application.

Master thesis

Master in Research and Innovation

Inferring programs structure from an execution trace

Author

Juan Francisco Martínez Vera

Supervisor

Jesús Labarta Mancho

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)

Refinement

Even if it **works pretty well** in general.
Some problems **have been identified**.

1. **Cluster aliasing** when two different loops behaves similarly enough over our defined space
2. **Hidden superloop** when not detected superloop prevents from a good structure detection.
3. **Cluster split** when MPI calls belonging to the same loop behaves in a different way.

All three **have been solved by looking to timestamps**.

Proposal modifications (ii)

Cluster aliasing

Keynote

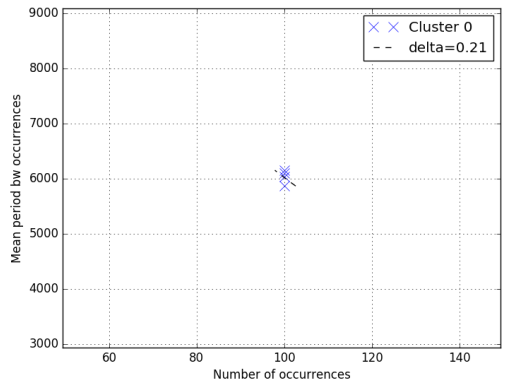
Aliasing can be detected and solved **looking at timestamps**.

for $1 to 100$

do $\begin{cases} MPI\_A() \\ MPI\_B() \end{cases}$

for $1 to 100$

do $\begin{cases} MPI\_C() \\ MPI\_D() \end{cases}$



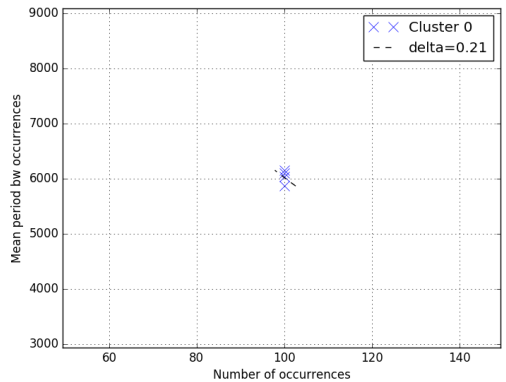
Proposal modifications (ii)

Cluster aliasing

Keynote

Aliasing can be detected and solved **looking at timestamps**.

for $1 to 100$
do $\begin{cases} MPI\_A() \\ MPI\_B() \end{cases}$
for $1 to 100$
do $\begin{cases} MPI\_C() \\ MPI\_D() \end{cases}$



MPI_A	1	3
MPI_B	2	4
MPI_C	5	7
MPI_D	6	8

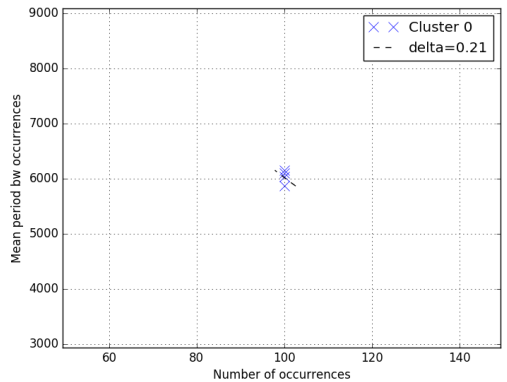
Proposal modifications (ii)

Cluster aliasing

Keynote

Aliasing can be detected and solved **looking at timestamps**.

for $1 to 100$
  do $\begin{cases} MPI\_A() \\ MPI\_B() \end{cases}$
for $1 to 100$
  do $\begin{cases} MPI\_C() \\ MPI\_D() \end{cases}$



MPI_A	1	3
MPI_B	2	4
MPI_C	5	7
MPI_D	6	8

MPI_A	1	3		
MPI_B	2	4		
MPI_C			5	7
MPI_D			6	8

Proposal modifications

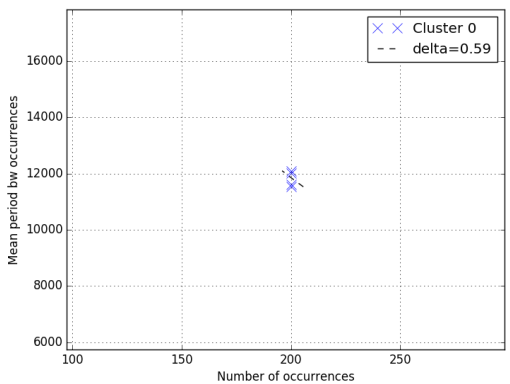
Hidden superloop

Keynote

Similarly than with aliasing, **looking at timestamps**.

for $1 to 100$

do {
- **for** $1 to 2$ **do** $\{MPI\_A()$
- **for** $1 to 2$ **do** $\{MPI\_B()$



Proposal modifications

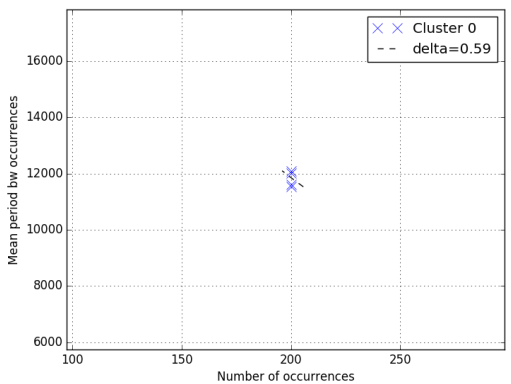
Hidden superloop

Keynote

Similarly than with aliasing, **looking at timestamps**.

for $1 to 100$

do {
- **for** $1 to 2$ **do** { $MPI\_A()$
- **for** $1 to 2$ **do** { $MPI\_B()$



MPI_A	1	2	5	6
MPI_B	3	4	7	8

Proposal modifications

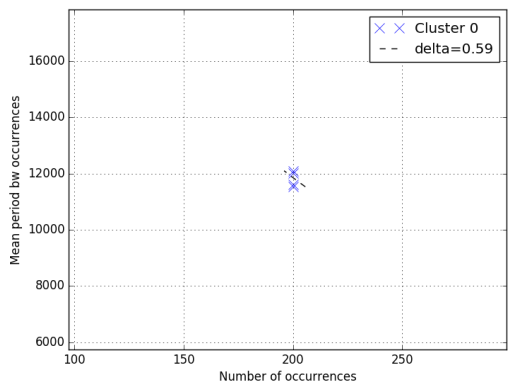
Hidden superloop

Keynote

Similarly than with aliasing, **looking at timestamps**.

for $1 to 100$

do {
- **for** $1 to 2$ **do** $\{MPI\_A()$
- **for** $1 to 2$ **do** $\{MPI\_B()$



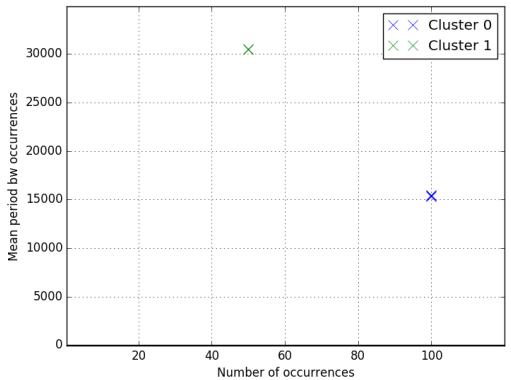
MPI_A	1	2	5	6
MPI_B	3	4	7	8

MPI_A	1	2			5	6		
MPI_B			3	4			7	8

Proposal modifications

Cluster split

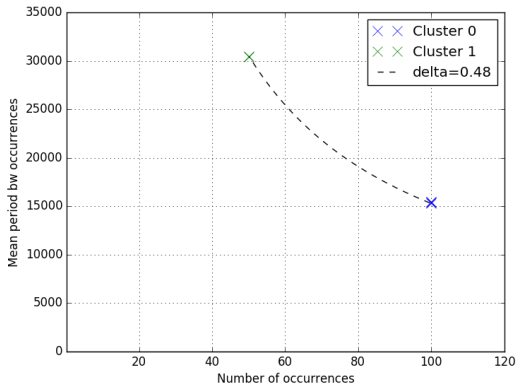
$$\textbf{for } i = 1 \text{ to } 10$$
$$\textbf{do} \begin{cases} MPI\_B() \\ \textbf{if } isPair(i) \\ \quad \textbf{then } MPI\_A() \\ MPI\_B() \end{cases}$$



Proposal modifications

Cluster split

$$\textbf{for } i = 1 \text{ to } 10 \quad \textbf{do} \begin{cases} MPI\_B() \\ \textbf{if } isPair(i) \\ \quad \textbf{then } MPI\_A() \\ MPI\_B() \end{cases}$$



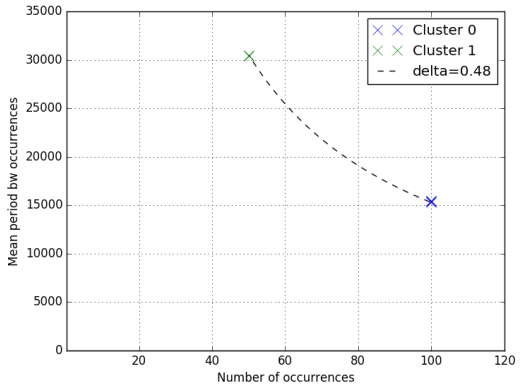
Whether MPI call is conditioned

- #repetitions and mean time bw. reps. change (in same proportion).

Proposal modifications

Cluster split

$$\textbf{for } i = 1 \text{ to } 10 \textbf{ do } \begin{cases} MPI\_B() \\ \textbf{if } isPair(i) \\ \quad \textbf{then } MPI\_A() \\ MPI\_B() \end{cases}$$



Whether MPI call is conditioned

- #repetitions and mean time bw. reps. change (in same proportion).

By "reverse loop merging" and times checking

FILE	LINE	PSEUDOCODE	E(TIME)	E(SIZE)	E(IPC)
	0	main()	-	-	-
		: FOR 1 TO 100	-	-	-
		: : IF rank in [1]	-	-	-
test-9.c	20	: : : MPI_Send(1:0)	5.01us	4.0B	1.1
		: : IF rank in [0]	-	-	-
test-9.c	22	: : : MPI_Recv()	5.3us	-	1.03
test-9.c	24	: : 50.0% => MPI_Barrier(CommId:1)	5.99us	-	0.97
		: : IF rank in [1]	-	-	-
test-9.c	26	: : : MPI_Send(1:0)	4.9us	4.0B	1.14
		: : IF rank in [0]	-	-	-
test-9.c	28	: : : MPI_Recv()	5.23us	-	1.1
		: END LOOP	-	-	-

Data analysis

Data acquisition (i)

Desired data...

- PCA[2]: Set of observations with set of features, i.e. Set of loops with with **iteration-level aggregated features**.
- Variable Importance Same data as in production traces but labeled, i.e. textbfWith information to what loop every MPI call belongs to.

How to obtain it

1. The needed information is not in trace so ...
2. **Manually inject monitors** to the source code
   - Very tough and error prone for huge source codes
3. Alternatively use source-to-source compiler to **do it utomaticaly**
   - Mercurium

[2] Principal component analysis

Data analysis

Data acquisition (i)

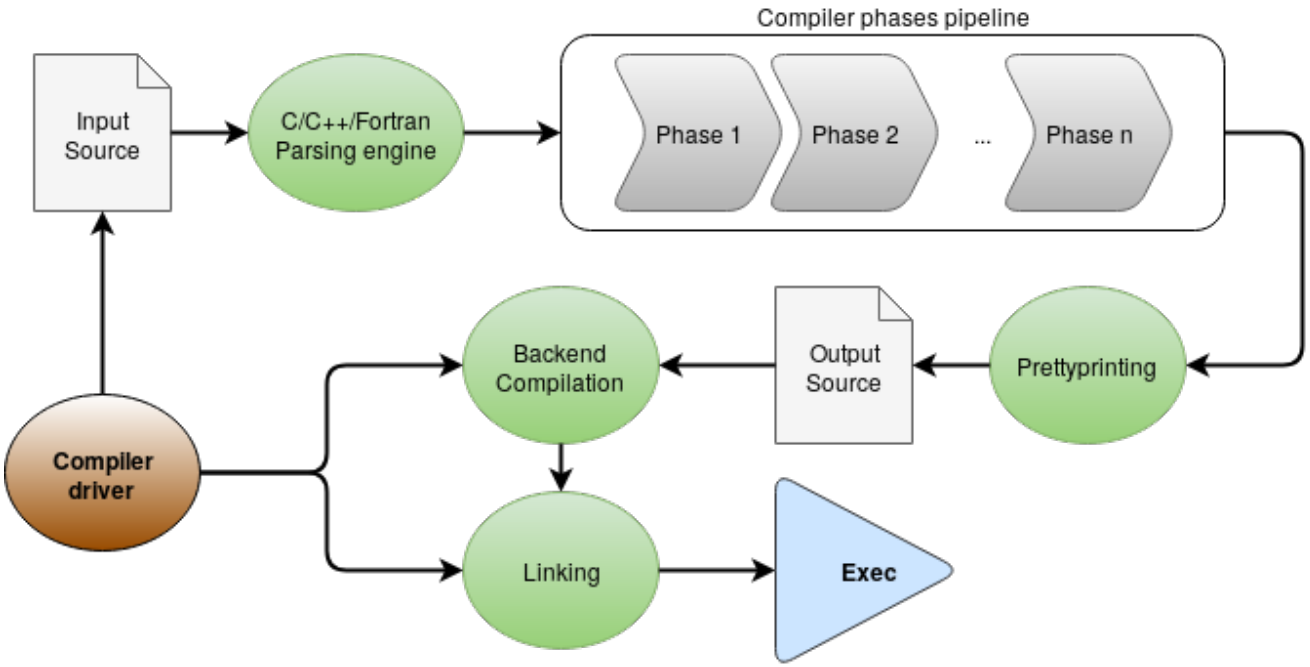


Figure: Mercurium internals overview

Data analysis

Data acquisition (ii)

Algorithm 11.1: $\mathrm{PCA}()$

MLoopInit($loop_{line}, loop_{file}$)
for $i \in I$
do {
  MIterInit(*chance*)
  ...
  MIterFini()
}
MLoopFini()

- MLoopInit and MLoopFini fire loops boundaries
- MIterInit and MiterFini fire iteration boundaries with hwc information o trace

Data analysis

Data acquisition (ii)

Algorithm 11.3: PCA()

$MLoopInit(loop_{line}, loop_{file})$
for $i \in I$
do {
  $MIterInit(chance)$
  ...
  $MIterFini()$
}
$MLoopFini()$

- MLoopInit and MLoopFini fire loops boundaries
- MIterInit and MiterFini fire iteration boundaries with hwc information o trace

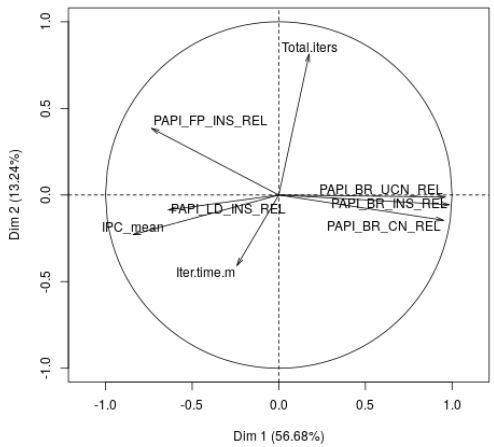
Algorithm 11.4: VarImp()

$MLoopInit(loop_{line}, loop_{file})$
for $i \in I$
do {
  ...
  $MBefCall()$
  $MPI\_Call()$
  $MPIAftCall()$
  ...
}
$MLoopFini()$

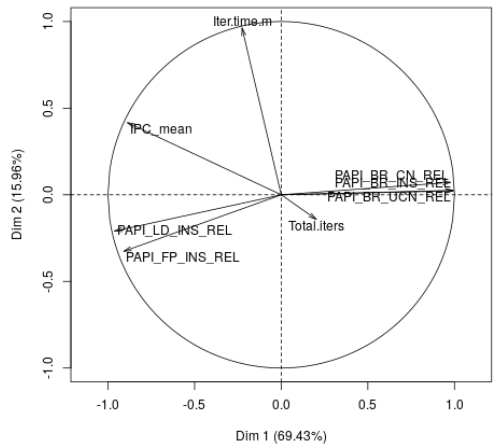
- MLoopInit and MLoopFini keep track entry/exit loops
- MPIBefCall and MPIAftCall fire loop id

Data analysis

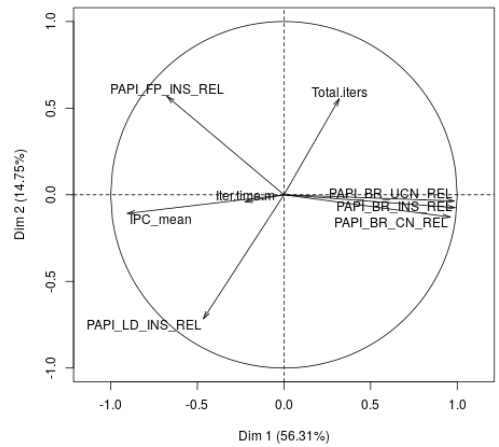
Analysis of results (i): **PCA**



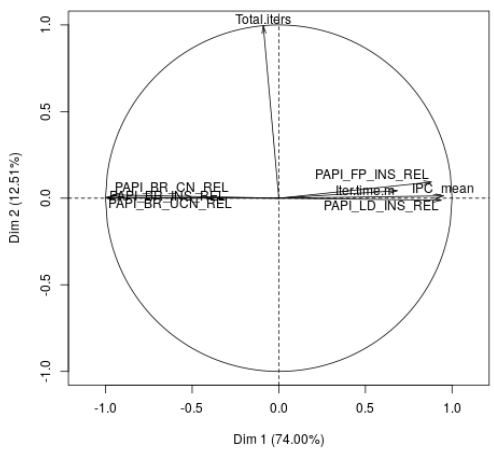
BT



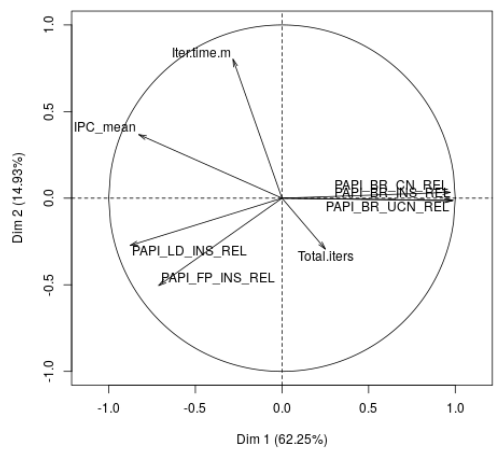
MG



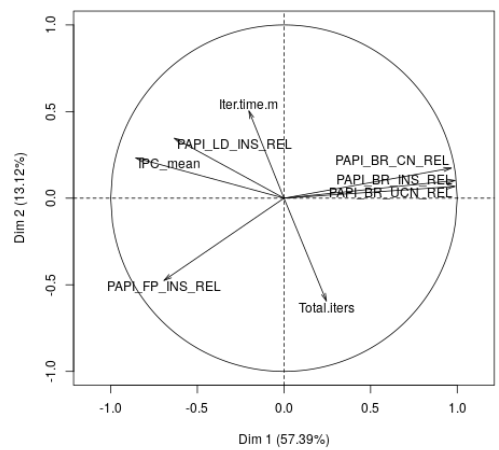
SP



CG



FT



LU

Results

Lulesh 2.0 (iii)

```
|lulesh.cc          |2773|: : TimeIncrement()                          |-          |-              |-          |
|lulesh.cc          | 214|: : : [~ computation ~]                      |56.63ms    |-              |2.56       |
|lulesh.cc          | 214|: : : 96.6% => MPI_Allreduce(CommId:1)       |35.32us    |8.0B/8.0B      |1.57       |
|lulesh.cc          |2774|: : LagrangeLeapFrog()                       |-          |-              |-          |
|lulesh.cc          |2648|: : : LagrangeNodal()                        |-          |-              |-          |
|lulesh.cc          |1263|: : : : CalcForceForNodes()                  |-          |-              |-          |
|lulesh.cc          |1137|: : : : : CommRecv()                         |-          |-              |-          |
|lulesh-comm.cc     | 101|: : : : : : MPI_Comm_rank(0:)                |10.27us    |-              |0.28       |
|lulesh-comm.cc     | 119|: : : : : : MPI_Irecv(0:25)                  |10.51us    |22.52KB        |0.55       |
|lulesh-comm.cc     | 137|: : : : : : MPI_Irecv(0:5)                   |8.67us     |22.52KB        |0.96       |
|lulesh-comm.cc     | 155|: : : : : : MPI_Irecv(0:1)                   |8.43us     |22.52KB        |1.08       |
|lulesh-comm.cc     | 192|: : : : : : MPI_Irecv(0:6)                   |8.8us      |744.0B         |1.08       |
|lulesh-comm.cc     | 201|: : : : : : MPI_Irecv(0:30)                  |8.54us     |744.0B         |1.25       |
|lulesh-comm.cc     | 210|: : : : : : MPI_Irecv(0:26)                  |8.29us     |744.0B         |1.25       |
|lulesh-comm.cc     | 345|: : : : : : MPI_Irecv(0:31)                  |8.32us     |24.0B          |0.9        |
|lulesh.cc          |1158|: : : : : CommSend()                         |-          |-              |-          |
|lulesh-comm.cc     | 401|: : : : : : [~ computation ~]                |85.2ms     |-              |2.48       |
|lulesh-comm.cc     | 401|: : : : : : MPI_Comm_rank(0:)                |20.08us    |-              |1.59       |
```

Table: Total MPI calls

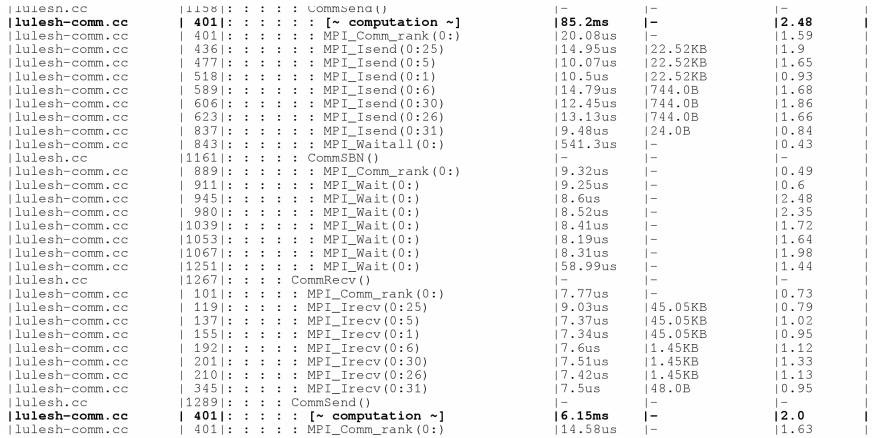
	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Rank 0	300	510	510	90	29	270

Table: MPI calls counts by communication phases

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Comm phase 1	0	7	0	0	1	1
Comm phase 2	7	7	7	1	0	3
Comm phase 3	0	0	7	1	0	2
Comm phase 4	0	3	0	0	0	1
Comm phase 5	3	0	3	1	0	2

Results

Lulesh 2.0 (iii)



```
|lulesh.cc          |1158|: : : : : CommSend()                  |-          |-          |-          |
|lulesh-comm.cc     | 401|: : : : : : [~ computation ~]         |85.2ms     |-          |2.48       |
|lulesh-comm.cc     | 401|: : : : : : MPI_Comm_rank(0:)         |20.08us    |-          |1.59       |
|lulesh-comm.cc     | 436|: : : : : : MPI_Isend(0:25)           |14.95us    |22.52KB    |1.9        |
|lulesh-comm.cc     | 477|: : : : : : MPI_Isend(0:5)            |10.07us    |22.52KB    |1.65       |
|lulesh-comm.cc     | 518|: : : : : : MPI_Isend(0:1)            |10.5us     |22.52KB    |0.93       |
|lulesh-comm.cc     | 589|: : : : : : MPI_Isend(0:6)            |14.79us    |744.0B     |1.68       |
|lulesh-comm.cc     | 606|: : : : : : MPI_Isend(0:30)           |12.45us    |744.0B     |1.86       |
|lulesh-comm.cc     | 623|: : : : : : MPI_Isend(0:26)           |13.13us    |744.0B     |1.66       |
|lulesh-comm.cc     | 837|: : : : : : MPI_Isend(0:31)           |9.48us     |24.0B      |0.84       |
|lulesh-comm.cc     | 843|: : : : : : MPI_Waitall(0:)           |541.3us    |-          |0.43       |
|lulesh.cc          |1161|: : : : : CommSBN()                   |-          |-          |-          |
|lulesh-comm.cc     | 889|: : : : : : MPI_Comm_rank(0:)         |9.32us     |-          |0.49       |
|lulesh-comm.cc     | 911|: : : : : : MPI_Wait(0:)              |9.25us     |-          |0.6        |
|lulesh-comm.cc     | 945|: : : : : : MPI_Wait(0:)              |8.6us      |-          |2.48       |
|lulesh-comm.cc     | 980|: : : : : : MPI_Wait(0:)              |8.52us     |-          |2.35       |
|lulesh-comm.cc     |1039|: : : : : : MPI_Wait(0:)              |8.41us     |-          |1.72       |
|lulesh-comm.cc     |1053|: : : : : : MPI_Wait(0:)              |8.19us     |-          |1.64       |
|lulesh-comm.cc     |1067|: : : : : : MPI_Wait(0:)              |8.31us     |-          |1.98       |
|lulesh-comm.cc     |1251|: : : : : : MPI_Wait(0:)              |58.99us    |-          |1.44       |
|lulesh.cc          |1267|: : : : CommRecv()                    |-          |-          |-          |
|lulesh-comm.cc     | 101|: : : : : MPI_Comm_rank(0:)           |7.77us     |-          |0.73       |
|lulesh-comm.cc     | 119|: : : : : MPI_Irecv(0:25)             |9.03us     |45.05KB    |0.79       |
|lulesh-comm.cc     | 137|: : : : : MPI_Irecv(0:5)              |7.37us     |45.05KB    |1.02       |
|lulesh-comm.cc     | 155|: : : : : MPI_Irecv(0:1)              |7.34us     |45.05KB    |0.95       |
|lulesh-comm.cc     | 192|: : : : : MPI_Irecv(0:6)              |7.6us      |1.45KB     |1.12       |
|lulesh-comm.cc     | 201|: : : : : MPI_Irecv(0:30)             |7.51us     |1.45KB     |1.33       |
|lulesh-comm.cc     | 210|: : : : : MPI_Irecv(0:26)             |7.42us     |1.45KB     |1.13       |
|lulesh-comm.cc     | 345|: : : : : MPI_Irecv(0:31)             |7.5us      |48.0B      |0.95       |
|lulesh.cc          |1289|: : : : CommSend()                    |-          |-          |-          |
|lulesh-comm.cc     | 401|: : : : : [~ computation ~]           |6.15ms     |-          |2.0        |
|lulesh-comm.cc     | 401|: : : : : MPI_Comm_rank(0:)           |14.58us    |-          |1.63       |
```

Table: Total MPI calls

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Rank 0	300	510	510	90	29	270

Table: MPI calls counts by communication phases

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Comm phase 1	0	7	0	0	1	1
Comm phase 2	7	7	7	1	0	3
Comm phase 3	0	0	7	1	0	2
Comm phase 4	0	3	0	0	0	1
Comm phase 5	3	0	3	1	0	2

Results

Lulesh 2.0 (iii)

```
|lulesh.cc          |1289|: : : : CommSend()                  |-          |-  |-          |
|lulesh-comm.cc     | 401|: : : : : [~ computation ~]         |6.15ms     |-  |2.0        |
|lulesh-comm.cc     | 401|: : : : : MPI_Comm_rank(0:)         |14.58us    |-  |1.63       |
|lulesh-comm.cc     | 843|: : : : : MPI_Waitall(0:)           |10.26us    |-  |0.35       |
|lulesh.cc          |1292|: : : : CommSyncPosVel()            |-          |-  |-          |
|lulesh-comm.cc     |1310|: : : : : MPI_Comm_rank(0:)         |22.14us    |-  |0.73       |
|lulesh-comm.cc     |1332|: : : : : MPI_Wait(0:)              |274.73us   |-  |0.79       |
|lulesh-comm.cc     |1366|: : : : : MPI_Wait(0:)              |72.95us    |-  |2.46       |
|lulesh-comm.cc     |1402|: : : : : MPI_Wait(0:)              |23.24us    |-  |2.3        |
|lulesh-comm.cc     |1461|: : : : : MPI_Wait(0:)              |198.36us   |-  |1.79       |
|lulesh-comm.cc     |1475|: : : : : MPI_Wait(0:)              |7.83us     |-  |1.83       |
|lulesh-comm.cc     |1489|: : : : : MPI_Wait(0:)              |7.75us     |-  |2.08       |
|lulesh-comm.cc     |1674|: : : : : MPI_Wait(0:)              |333.07us   |-  |1.73       |
|lulesh.cc          |2656|: : : LagrangeElements()            |-          |-  |-          |
|lulesh.cc          |2461|: : : : CalcQForElems()             |-          |-  |-          |
|lulesh.cc          |1992|: : : : : CommRecv()                |-          |-  |-          |
|lulesh-comm.cc     | 101|: : : : : : [~ computation ~]       |29.32ms    |-  |2.25       |
|lulesh-comm.cc     | 101|: : : : : : MPI_Comm_rank(0:)       |18.76us    |-  |1.39       |
```

Table: Total MPI calls

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Rank 0	300	510	510	90	29	270

Table: MPI calls counts by communication phases

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Comm phase 1	0	7	0	0	1	1
Comm phase 2	7	7	7	1	0	3
Comm phase 3	0	0	7	1	0	2
Comm phase 4	0	3	0	0	0	1
Comm phase 5	3	0	3	1	0	2

Results

Lulesh 2.0 (iii)

```
|lulesh.cc          |1992|: : : : : CommRecv()                    |          |               |          |
|lulesh-comm.cc     | 101|: : : : : : [~ computation ~]           |29.32ms   |-              |2.25      |
|lulesh-comm.cc     | 101|: : : : : : MPI_Comm_rank(0:)           |18.76us   |-              |1.39      |
|lulesh-comm.cc     | 119|: : : : : : MPI_Irecv(0:25)             |12.61us   |21.09KB        |0.43      |
|lulesh-comm.cc     | 137|: : : : : : MPI_Irecv(0:5)              |9.23us    |21.09KB        |1.01      |
|lulesh-comm.cc     | 155|: : : : : : MPI_Irecv(0:1)              |8.79us    |21.09KB        |1.17      |
|lulesh.cc          |2010|: : : : : CommSend()                    |-         |-              |-         |
|lulesh-comm.cc     | 401|: : : : : : [~ computation ~]           |15.35ms   |-              |1.91      |
|lulesh-comm.cc     | 401|: : : : : : MPI_Comm_rank(0:)           |15.24us   |-              |1.52      |
|lulesh-comm.cc     | 436|: : : : : : MPI_Isend(0:25)             |13.14us   |21.09KB        |1.9       |
|lulesh-comm.cc     | 477|: : : : : : MPI_Isend(0:5)              |10.28us   |21.09KB        |1.89      |
|lulesh-comm.cc     | 518|: : : : : : MPI_Isend(0:1)              |10.57us   |21.09KB        |1.26      |
|lulesh-comm.cc     | 843|: : : : : : MPI_Waitall(0:)             |291.68us  |-              |0.45      |
|lulesh.cc          |2014|: : : : : CommMonoQ()                   |-         |-              |-         |
|lulesh-comm.cc     |1734|: : : : : : MPI_Comm_rank(0:)           |9.51us    |-              |0.45      |
|lulesh-comm.cc     |1757|: : : : : : MPI_Wait(0:)                |8.88us    |-              |0.81      |
|lulesh-comm.cc     |1791|: : : : : : MPI_Wait(0:)                |8.78us    |-              |2.04      |
|lulesh-comm.cc     |1824|: : : : : : MPI_Wait(0:)                |8.53us    |-              |2.13      |
|                   |    |: END LOOP                              |-         |-              |-         |
+-------------------+----+----------------------------------------+----------+---------------+----------+
```

Table: Total MPI calls

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Rank 0	300	510	510	90	29	270

Table: MPI calls counts by communication phases

	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
Comm phase 1	0	7	0	0	1	1
Comm phase 2	7	7	7	1	0	3
Comm phase 3	0	0	7	1	0	2
Comm phase 4	0	3	0	0	0	1
Comm phase 5	3	0	3	1	0	2