# Actors to Threads Mapping Technique for JVM-based Actor Frameworks

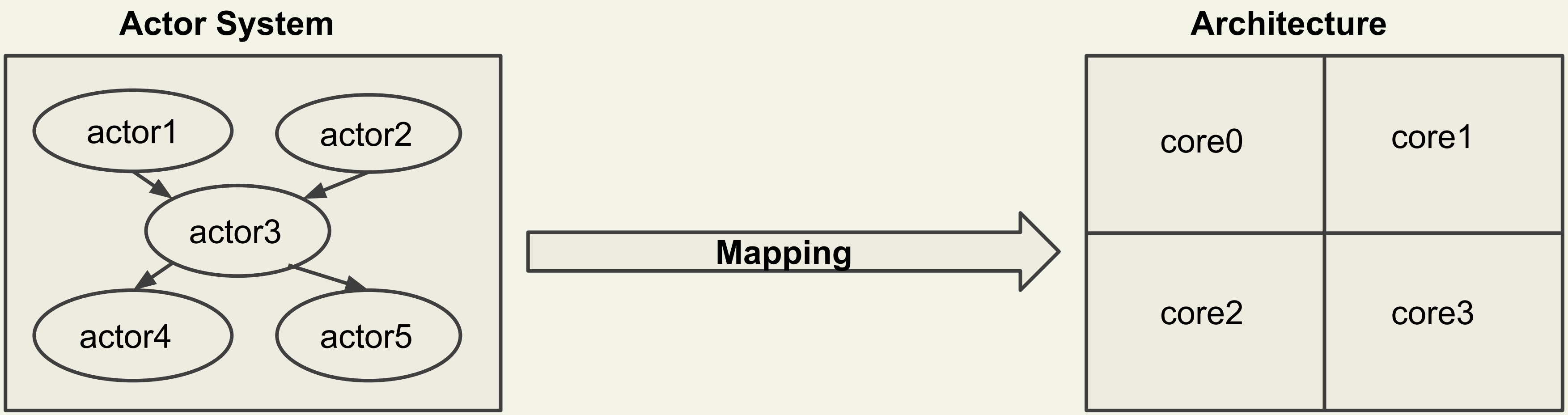
Ganesha Upadhyaya and Hridesh Rajan

{ganeshau,hridesh}@iastate.edu

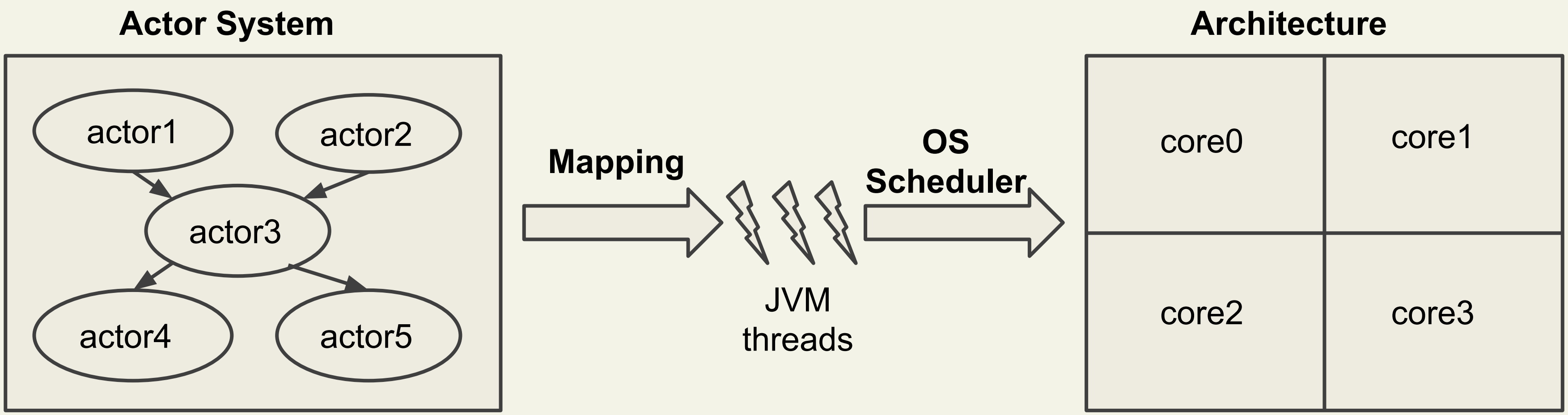
Iowa State University

This work was supported in part by the NSF grants CCF- 08-46059, CCF-11-17937, and CCF-14-23370.

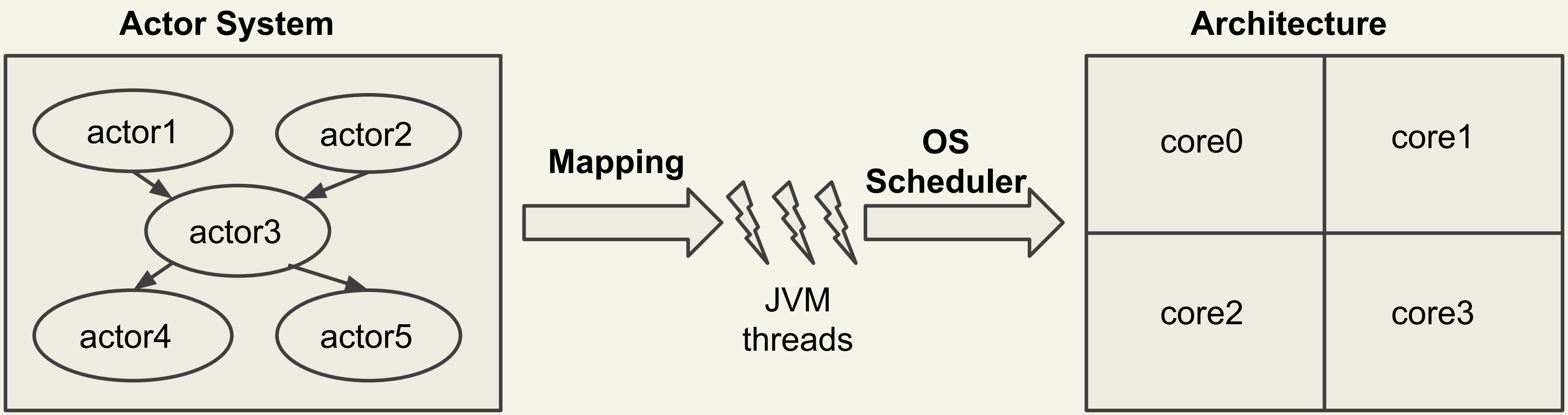
# Problem


Actor System
actor1
actor2
actor3
actor4
actor5
Mapping
Architecture
core0
core1
core2
core3

# Problem

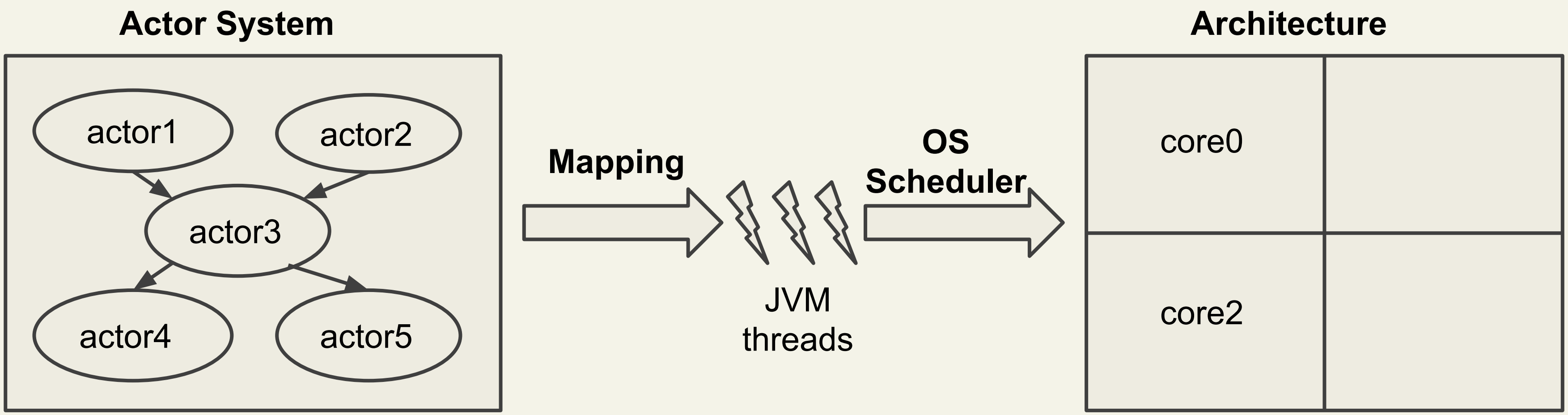

Actor System
actor1
actor2
actor3
actor4
actor5
Mapping
JVM threads
OS Scheduler
Architecture
core0
core1
core2
core3

# Problem




PROBLEM: **Mapping Actors to JVM threads**

# Problem

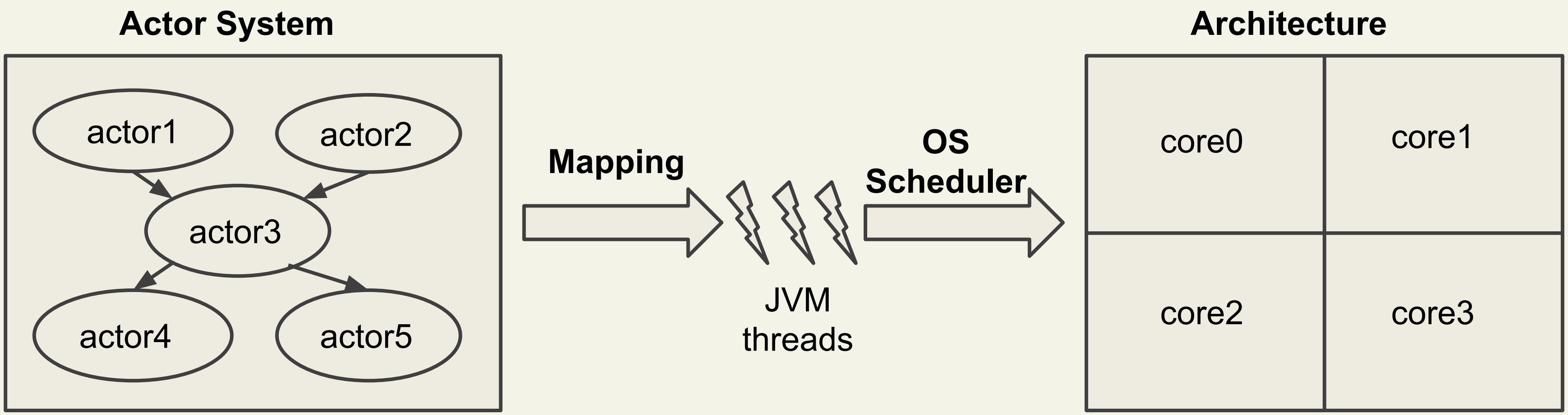



**PROBLEM: Mapping Actors to JVM threads**

- More Actors than JVM threads,
- Actors are short-lived [1],
- OS Scheduler maps JVM threads to Cores,

[1] Francesquini, Emilio, Alfredo Goldman, and Jean-François Mehaut. "Improving the performance of actor model runtime environments on multicore and manycore platforms." Proceedings of AGERE, 2013.

# Problem



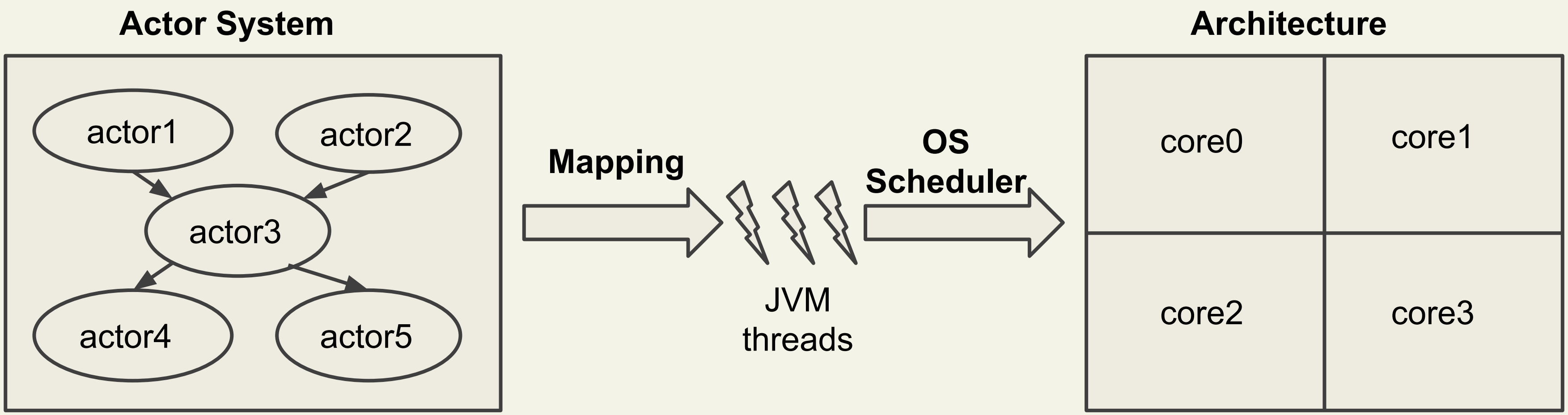

PROBLEM: **Mapping Actors to JVM threads**

- More Actors than JVM threads,
- Actors are short-lived [1],
- OS Scheduler maps JVM threads to Cores,

INTUITION: Actor characteristics and Communication behaviours could be used to decide the mapping

[1] Francesquini, Emilio, Alfredo Goldman, and Jean-François Mehaut. "Improving the performance of actor model runtime environments on multicore and manycore platforms." Proceedings of AGERE, 2013.

# Problem




PROBLEM: **Mapping Actors to JVM threads**

- More Actors than JVM threads,
- Actors are short-lived [1],
- OS Scheduler maps JVM threads to Cores,

INTUITION: Actor characteristics and Communication behaviours could be used to decide the mapping

SOLUTION: **Initial mapping of Actors to JVM threads**

[1] Francesquini, Emilio, Alfredo Goldman, and Jean-François Mehaut. "Improving the performance of actor model runtime environments on multicore and manycore platforms." Proceedings of AGERE, 2013.

# Outline

- Discuss JVM based actor frameworks
- Motivating examples
- Solution
- Illustrative example
- Evaluation & Results
- Limitations and Future Work.

# JVM-based actor frameworks

Akka | Scala Actors | SALSA

Kilim | Jetlang | ActorFoundry | Actors Guild

## Akka

- default
- pinned
- balancing
- calling-thread

## Kilim, Actor Foundry

- light-weight event-based actors,
- scheduler is a bundle composed of a thread-pool, scheduling policy, collection of runnable actors,
- scheduled in round-robin fashion

## SALSA

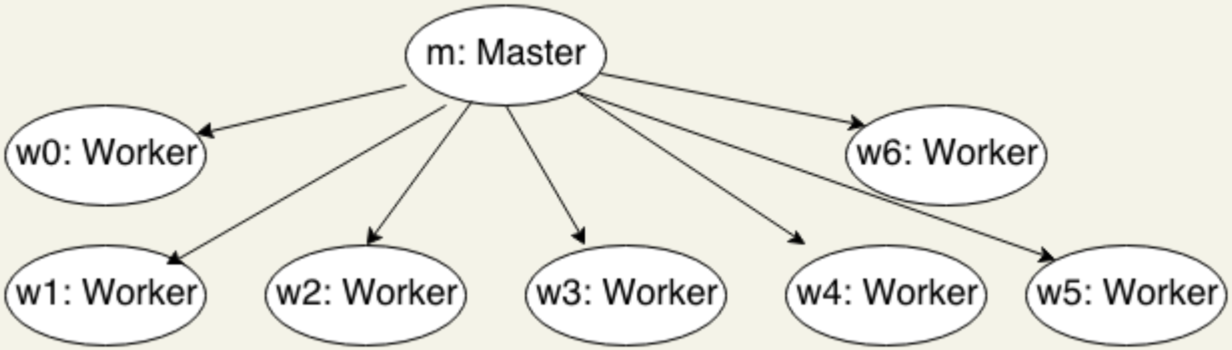
- heavy-weight (individual stage)
- light-weight (stage-sharing)
- each stage (actor) is a bundle of a msgQ and JVM thread

## Scala Actors, Actors Guild

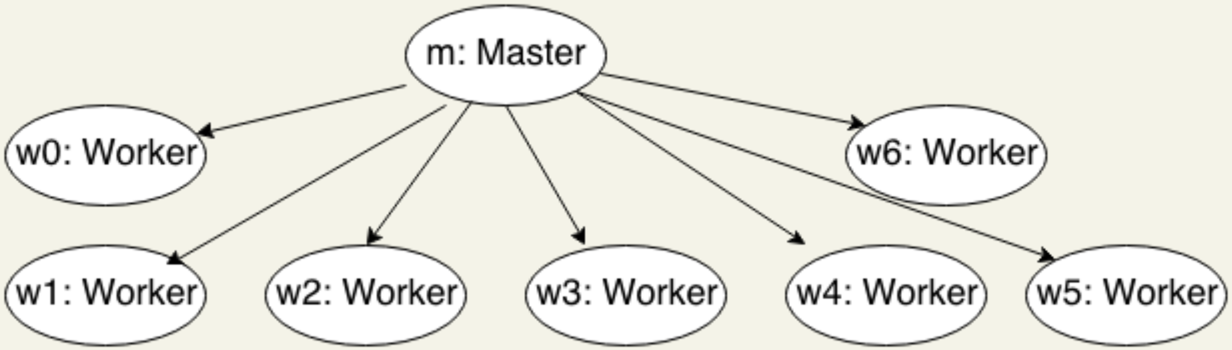
- thread-based
- event-based

- **start with default mappings,**
- **iteratively refine the mappings to achieve desired performance**

# Example1: Master-Worker (RayTracer)

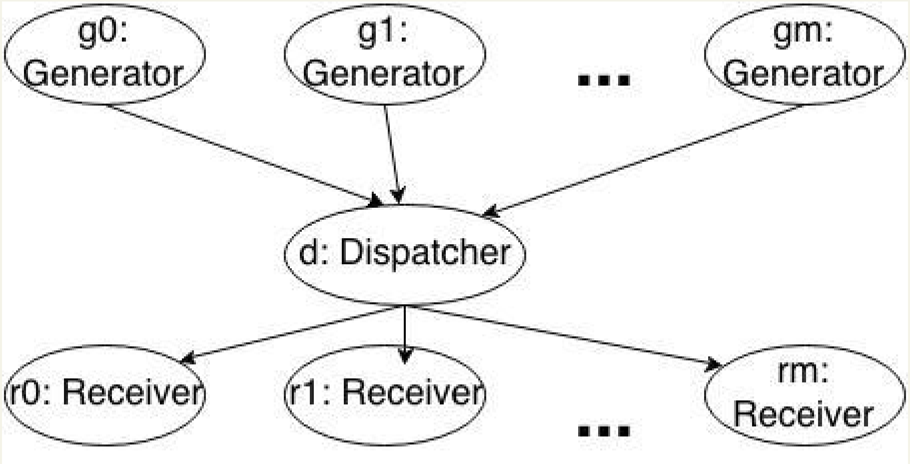


# Example1: Master-Worker (RayTracer)

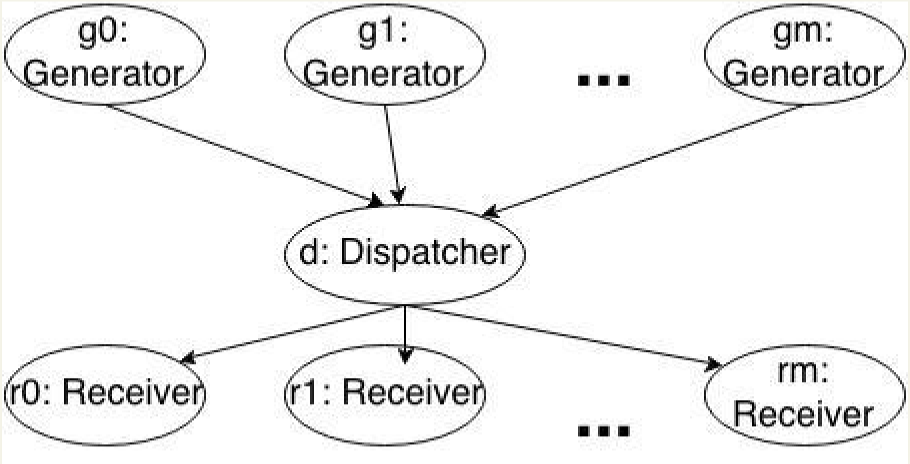



- ➢ easy to map actors to JVM threads,
- ➢ because actors perform independent computations,
- ➢ less interactions,
- ➢ data-parallel.

# Example2: BenchErl-Serialmsg

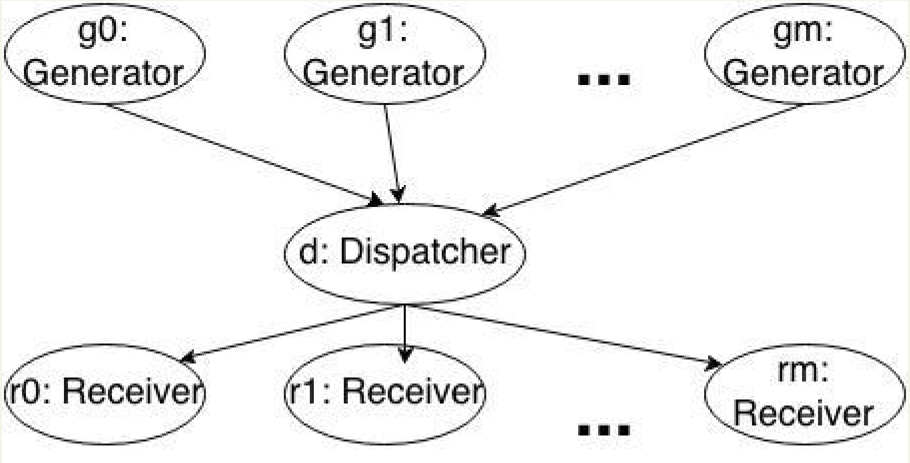


# Example2: BenchErl-Serialmsg




- m instances of ***Generator*** actor,
- m instances of ***Receiver*** actor,
- one ***Dispatcher*** actor,

# Example2: BenchErl-Serialmsg

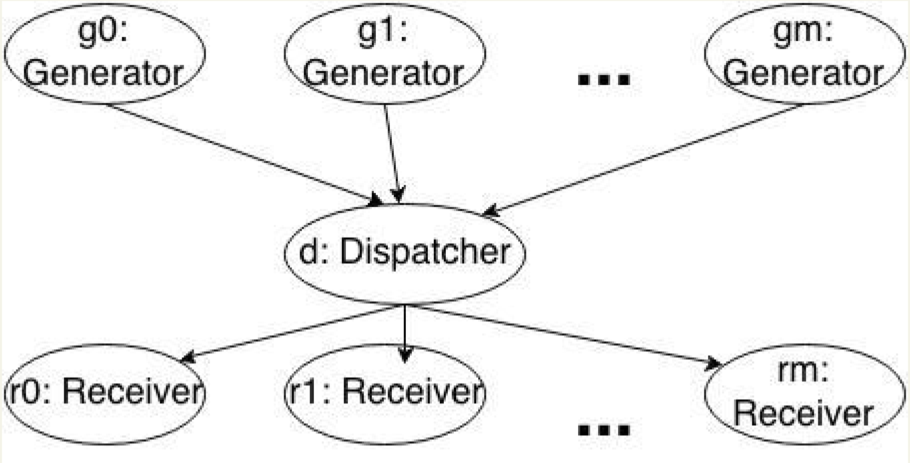



- m instances of ***Generator*** actor,
- m instances of ***Receiver*** actor,
- one ***Dispatcher*** actor,

## Intuitive Mapping Process

- start with a task-pool (size=#cores) & put all actors in there,
- looks like Dispatcher is a bottleneck,
- assign a thread-pool to Dispatcher,
- still not working, load-imbalance, *how do I do it?*

# Example2: BenchErl-Serialmsg




- m instances of ***Generator*** actor,
- m instances of ***Receiver*** actor,
- one ***Dispatcher*** actor,

## Intuitive Mapping Process

- start with a task-pool (size=#cores) & put all actors in there,
- looks like Dispatcher is a bottleneck,
- assign a thread-pool to Dispatcher,
- still not working, load-imbalance, *how do I do it?*

## Insight

- each generator communicates with receiver often through dispatcher,
- whole communication (g0 -> d -> r0) could be made uninterrupted

*http://letitcrash.com/post/40755146949/tuning-dispatchers-in-akka-applications*

# Observation: Analyzing Actor Applications

*Some aspects of actor applications can help to decide actors to JVM threads mapping*

# Observation: Analyzing Actor Applications

*Some aspects of actor applications can help to decide actors to JVM threads mapping*

- ***blocking***,
  - externally blocking behaviors using I/O, socket or database blocking primitives,

# Observation: Analyzing Actor Applications

*Some aspects of actor applications can help to decide actors to JVM threads mapping*

- ***blocking***,
  - externally blocking behaviors using I/O, socket or database blocking primitives,
- ***inherent parallelism***,
  - actors may use blocking send primitives and receive results or use asynchronous send primitives. Actors may or may not require the results immediately,

# Observation: Analyzing Actor Applications

*Some aspects of actor applications can help to decide actors to JVM threads mapping*

- ***blocking***,
  - externally blocking behaviors using I/O, socket or database blocking primitives,
- ***inherent parallelism***,
  - actors may use blocking send primitives and receive results or use asynchronous send primitives. Actors may or may not require the results immediately,
- ***communication behavior***,
  - leaf actor,
  - routing actor,
  - broadcast actor.
- ***computations***,

# Actor Characteristic Vector (cVector)

| BLK | STATE | PAR | COMM | CPU |
|---|---|---|---|---|

# Actor Characteristic Vector (cVector)

| | | | | |
|---|---|---|---|---|
| **BLK** | **STATE** | **PAR** | **COMM** | **CPU** |

- **BLK** = *{true, false}* represents blocking behavior,

# Actor Characteristic Vector (cVector)

| BLK | STATE | PAR | COMM | CPU |
| --- | --- | --- | --- | --- |

- **BLK** = *{true, false}* represents blocking behavior,
- **STATE** = *{true, false}* represents stateful/stateless behavior,

# Actor Characteristic Vector (cVector)

| BLK | STATE | PAR | COMM | CPU |
|---|---|---|---|---|

- **BLK** = *{true, false}* represents blocking behavior,
- **STATE** = *{true, false}* represents stateful/stateless behavior,
- **PAR** = *{low, med, high}* represents inherent parallelism,
  - *low*, if actor sends synchronous message and waits for the result, or consumes the result right-away,
  - *high*, if actor sends asynchronous message and does not require result,
  - *med*, otherwise.

# Actor Characteristic Vector (cVector)

| BLK | STATE | PAR | COMM | CPU |
|---|---|---|---|---|

- **BLK** = *{true, false}* represents blocking behavior,
- **STATE** = *{true, false}* represents stateful/stateless behavior,
- **PAR** = *{low, med, high}* represents inherent parallelism,
  - *low*, if actor sends synchronous message and waits for the result, or consumes the result right-away,
  - *high*, if actor sends asynchronous message and does not require result,
  - *med*, otherwise.
- **COMM** = *{low, med, high}* represents communication behavior,
  - *low*, does not send messages to other actors (leaf actor),
  - *med*, sends exactly one message for every message received (router actor),
  - *high*, sends more than one message (broadcast actor).

# Actor Characteristic Vector (cVector)

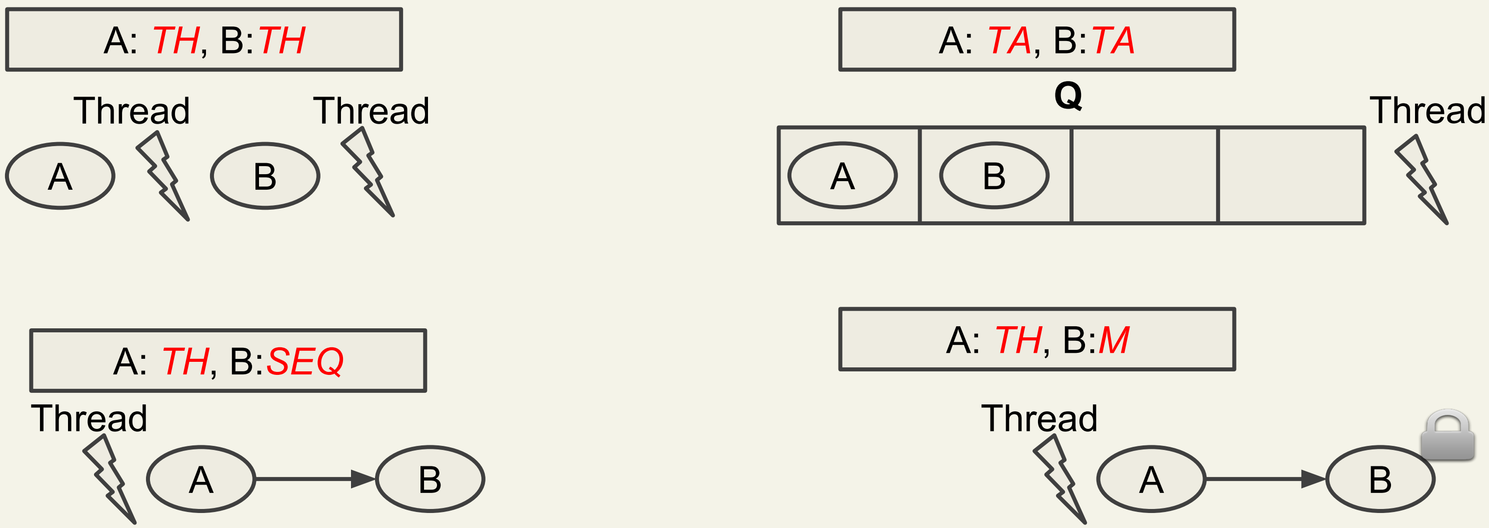
| BLK | STATE | PAR | COMM | CPU |
|---|---|---|---|---|

- **BLK** = *{true, false}* represents blocking behavior,
- **STATE** = *{true, false}* represents stateful/stateless behavior,
- **PAR** = *{low, med, high}* represents inherent parallelism,
  - *low*, if actor sends synchronous message and waits for the result, or consumes the result right-away,
  - *high*, if actor sends asynchronous message and does not require result,
  - *med*, otherwise.
- **COMM** = *{low, med, high}* represents communication behavior,
  - *low*, does not send messages to other actors (leaf actor),
  - *med*, sends exactly one message for every message received (router actor),
  - *high*, sends more than one message (broadcast actor).
- **CPU** = *{low, high}* represents computational workload of the actor,
  - *high*, when recursive, loops with unknown bounds, makes high cost library calls,
  - *low*, otherwise.

# Solution

- For mapping actors to threads,
  - we assign execution policy to actors,
- execution policy,
  - defines, how actor’s messages are processed?

# Execution Policies




*TH*: Thread *TA*: Task *SEQ*: Sequential *M*: Monitor

- **THREAD**, actor is assigned a dedicated thread,
- **TASK**, actor is assigned to a task-pool and the shared thread of the task-pool will process the messages,
- **SEQ/MONITOR**, calling actor thread itself

# Mapping Function

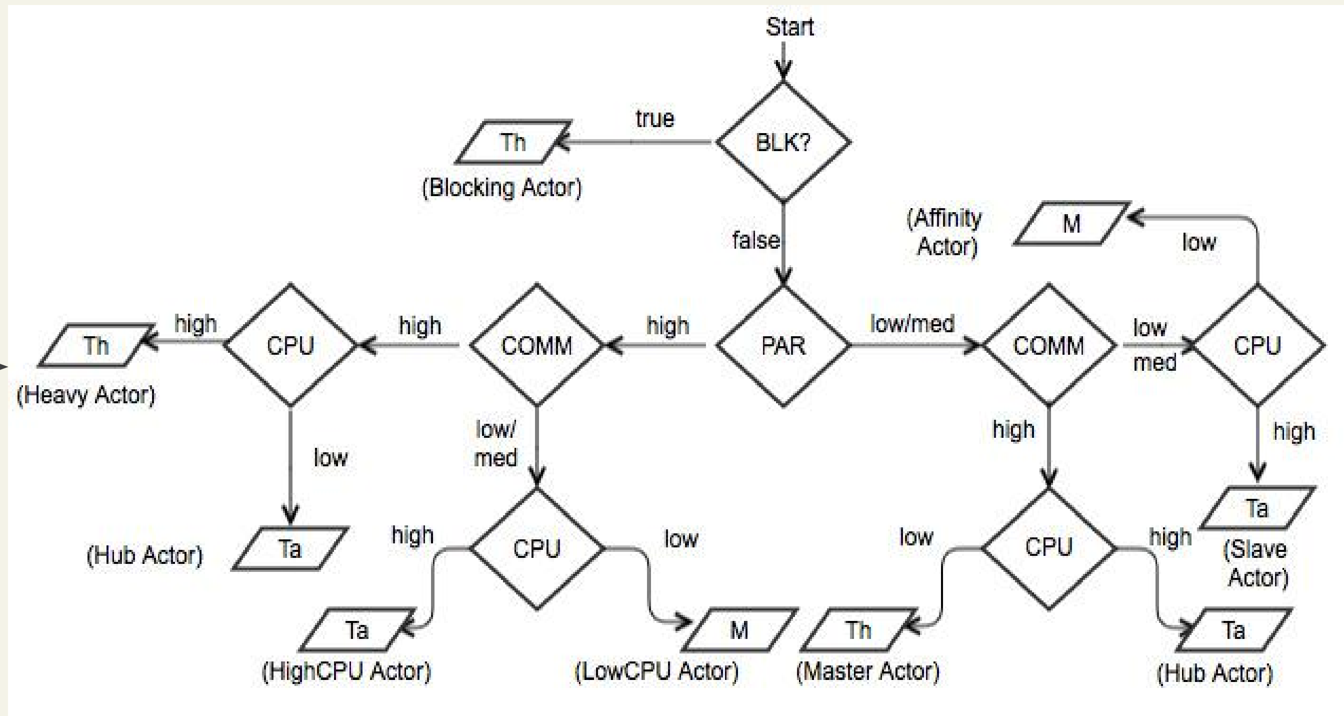
- **Actor Communication Graph (ACG)** is a directed graph $G(V,E)$ where,
  - $V = A0, A1, \ldots, An$ is a set of nodes, each node represents an actor,
  - $E$ is a set of edges $(Ai, Aj)$ for all $i,j$ such that there is a communication from $Ai$ to $Aj$.

- Mapping function $M\ (Ai \times P \times ACG) \longmapsto EP$ where,
  - $Ai$ is actor definition,
  - $P$ is the actor program,
  - $ACG$ is the actor communication graph,
  - EP = { *THREAD* | *TASK* | *SEQ* | *MONITOR* }

# Mapping Function: Flow diagram

Input

cVector




Output

Execution Policy

Figure: Flow diagram of our mapping function that assigns actors one of the four execution policies.

# Actor Characteristic Vector (cVector)

| Heuristics | BLK | STATE | PAR | COMM | CPU | Policy |
| --- | --- | --- | --- | --- | --- | --- |
| *Blocking Actors* | true | ⃠ | ⃠ | ⃠ | ⃠ | THREAD |

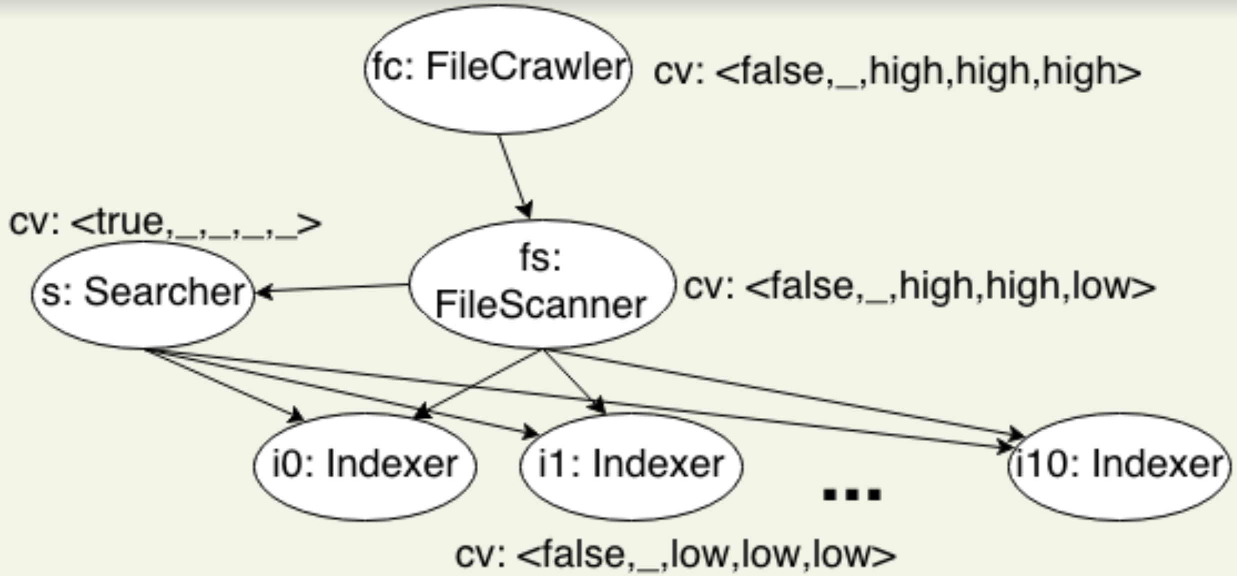
⃠ *means not used to make decision*

# Actor Characteristic Vector (cVector)

| **Heuristics** | BLK | STATE | PAR | COMM | CPU | **Policy** |
|---|---|---|---|---|---|---|
| *Blocking Actors* | true | ⊘ | ⊘ | ⊘ | ⊘ | THREAD |
| *Heavy Actors* | false | ⊘ | high | high | high | THREAD |
| HighCPU Actors | false | ⊘ | high | low | high | TASK |
| LowCPU Actors | false | ⊘ | high | low | low | MONITOR |
| Hub Actors | false | ⊘ | high | high | low | TASK |
| Affinity Actors | false | ⊘ | low/med | low/med | low | MONITOR |
| Master Actors | false | ⊘ | low/med | high | low | THREAD |
| Worker Actors | false | ⊘ | low/med | low/med | high | TASK |

⊘ *means not used to make decision*

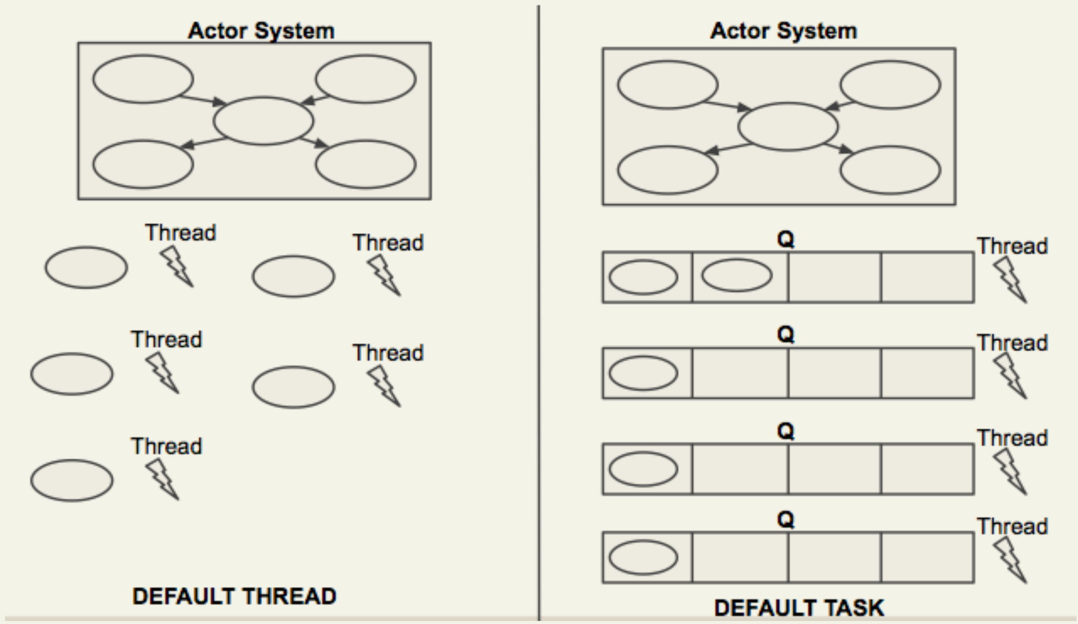
# An Example: FileSearch



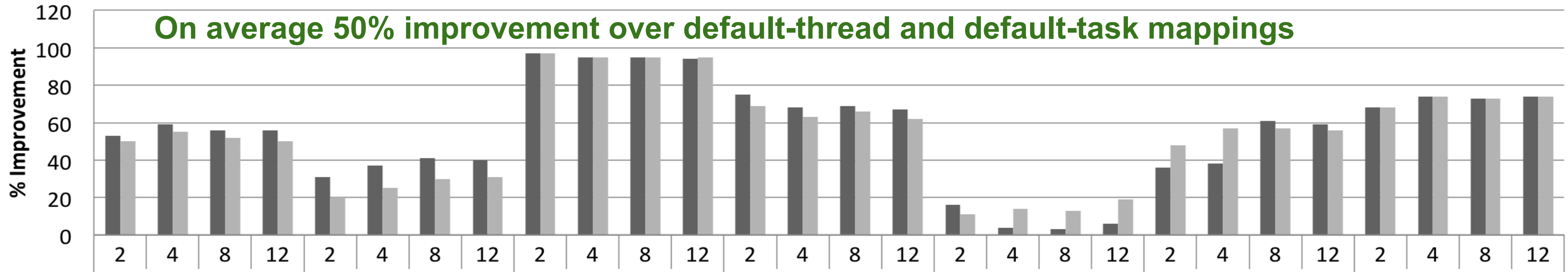

| Capsule | cVector | Policy |
|---|---|---|
| FileCrawler | <**false**,_,high,high,high> | *Thread* |
| FileScanner | <**false**,_,high,high,low> | *Task* |
| Indexer | <**false**,_,low,low,low> | *Monitor* |
| Searcher | <**true**,_,_,_,_> | *Thread* |

# Evaluation

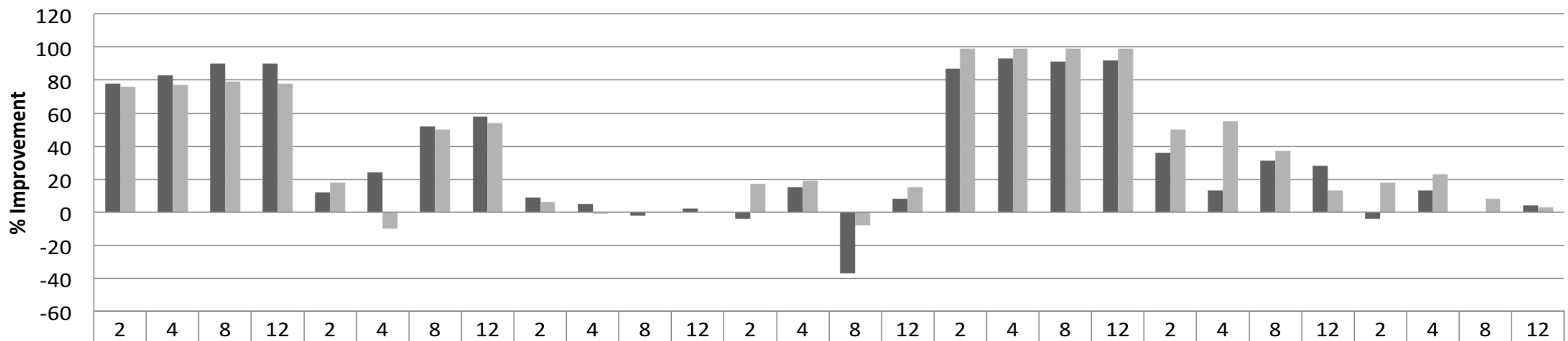
- Benchmark programs (14 total)
  - that exhibits data, task, and pipeline parallelism at coarse and fine granularities.
- Comparing against default-thread and default-task,
- Measured reduction in program runtime over default mappings on different core settings.



# Experimental Results



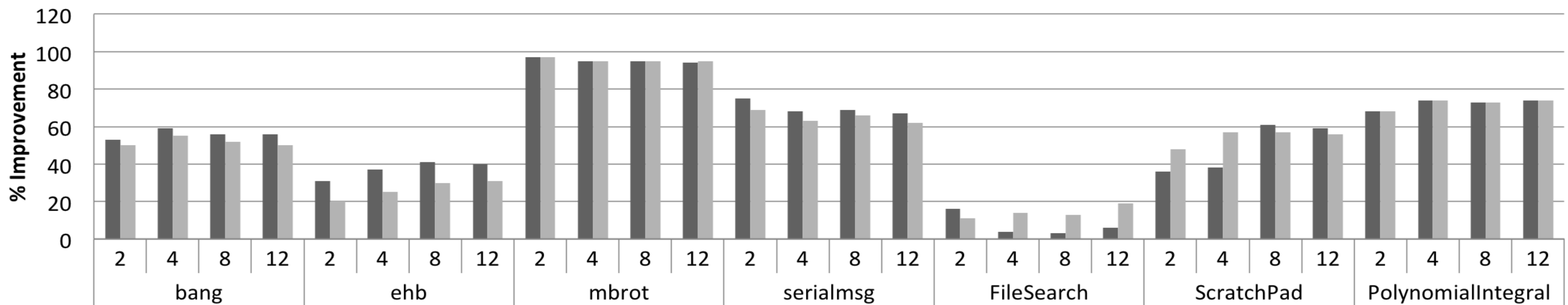

| | bang | | | | ehb | | | | mbrot | | | | serialmsg | | | | FileSearch | | | | ScratchPad | | | | PolynomialIntegral | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 53 | 59 | 56 | 56 | 31 | 37 | 41 | 40 | 97 | 95 | 95 | 94 | 75 | 68 | 69 | 67 | 16 | 4 | 3 | 6 | 36 | 38 | 61 | 59 | 68 | 74 | 73 | 74 |
| Ita | 50 | 55 | 52 | 50 | 20 | 25 | 30 | 31 | 97 | 95 | 95 | 95 | 69 | 63 | 66 | 62 | 11 | 14 | 13 | 19 | 48 | 57 | 57 | 56 | 68 | 74 | 73 | 74 |



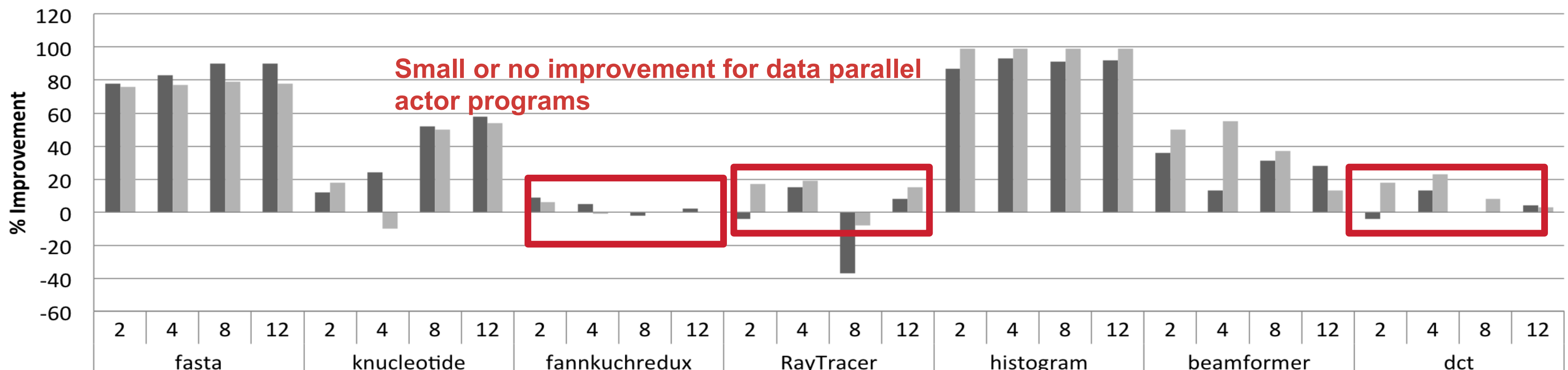

| | fasta | | | | knucleotide | | | | fannkuchredux | | | | RayTracer | | | | histogram | | | | beamformer | | | | dct | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 78 | 83 | 90 | 90 | 12 | 24 | 52 | 58 | 9 | 5 | -2 | 2 | -4 | 15 | -37 | 8 | 87 | 93 | 91 | 92 | 36 | 13 | 31 | 28 | -4 | 13 | 0 | 4 |
| Ita | 76 | 77 | 79 | 78 | 18 | -10 | 50 | 54 | 6 | -1 | 0 | 0 | 17 | 19 | -8 | 15 | 99 | 99 | 99 | 99 | 50 | 55 | 37 | 13 | 18 | 23 | 8 | 3 |

Figure: Results show Ith (improvement over default-thread mapping) and Ita (improvement over default-task mapping) for the benchmarks.

# Experimental Results



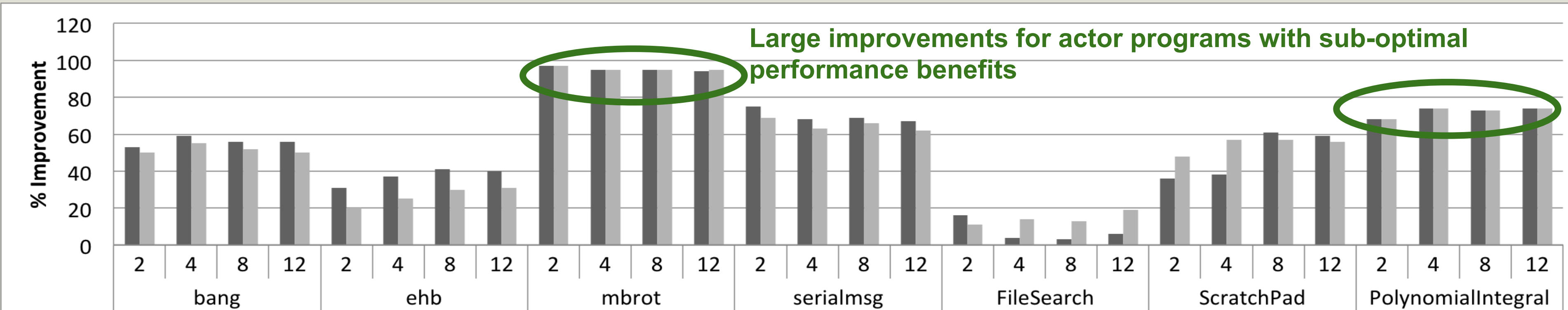

| | bang | | | | ehb | | | | mbrot | | | | serialmsg | | | | FileSearch | | | | ScratchPad | | | | PolynomialIntegral | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 53 | 59 | 56 | 56 | 31 | 37 | 41 | 40 | 97 | 95 | 95 | 94 | 75 | 68 | 69 | 67 | 16 | 4 | 3 | 6 | 36 | 38 | 61 | 59 | 68 | 74 | 73 | 74 |
| Ita | 50 | 55 | 52 | 50 | 20 | 25 | 30 | 31 | 97 | 95 | 95 | 95 | 69 | 63 | 66 | 62 | 11 | 14 | 13 | 19 | 48 | 57 | 57 | 56 | 68 | 74 | 73 | 74 |




| | fasta | | | | knucleotide | | | | fannkuchredux | | | | RayTracer | | | | histogram | | | | beamformer | | | | dct | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 78 | 83 | 90 | 90 | 12 | 24 | 52 | 58 | 9 | 5 | -2 | 2 | -4 | 15 | -37 | 8 | 87 | 93 | 91 | 92 | 36 | 13 | 31 | 28 | -4 | 13 | 0 | 4 |
| Ita | 76 | 77 | 79 | 78 | 18 | -10 | 50 | 54 | 6 | -1 | 0 | 0 | 17 | 19 | -8 | 15 | 99 | 99 | 99 | 99 | 50 | 55 | 37 | 13 | 18 | 23 | 8 | 3 |

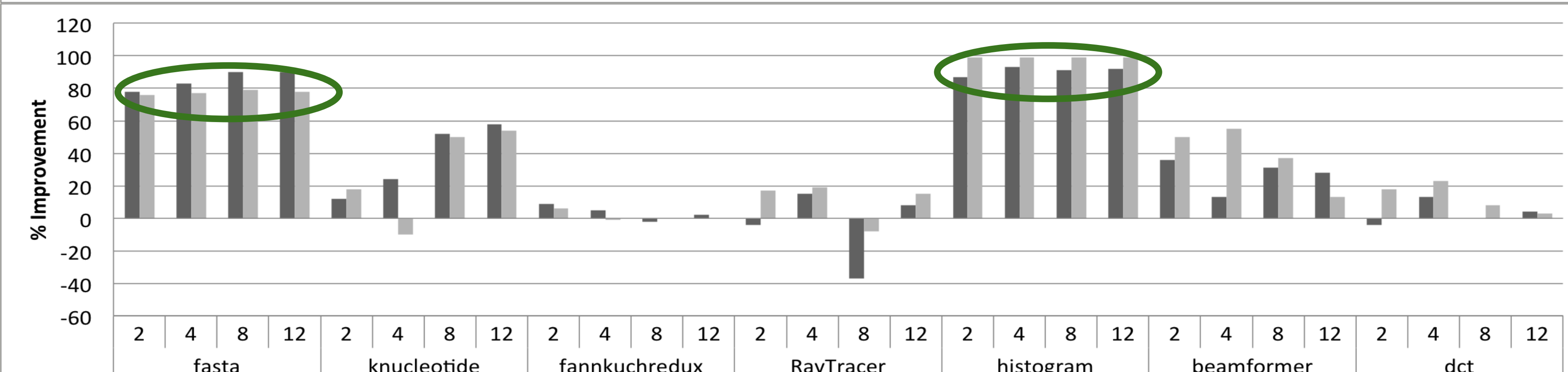
Figure: Results show Ith (improvement over default-thread mapping) and Ita (improvement over default-task mapping) for the benchmarks.

# Experimental Results




| | bang | | | | ehb | | | | mbrot | | | | serialmsg | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 53 | 59 | 56 | 56 | 31 | 37 | 41 | 40 | 97 | 95 | 95 | 94 | 75 | 68 | 69 | 67 |
| Ita | 50 | 55 | 52 | 50 | 20 | 25 | 30 | 31 | 97 | 95 | 95 | 95 | 69 | 63 | 66 | 62 |

| | FileSearch | | | | ScratchPad | | | | PolynomialIntegral | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 16 | 4 | 3 | 6 | 36 | 38 | 61 | 59 | 68 | 74 | 73 | 74 |
| Ita | 11 | 14 | 13 | 19 | 48 | 57 | 57 | 56 | 68 | 74 | 73 | 74 |




| | fasta | | | | knucleotide | | | | fannkuchredux | | | | RayTracer | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 78 | 83 | 90 | 90 | 12 | 24 | 52 | 58 | 9 | 5 | -2 | 2 | -4 | 15 | -37 | 8 |
| Ita | 76 | 77 | 79 | 78 | 18 | -10 | 50 | 54 | 6 | -1 | 0 | 0 | 17 | 19 | -8 | 15 |

| | histogram | | | | beamformer | | | | dct | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 87 | 93 | 91 | 92 | 36 | 13 | 31 | 28 | -4 | 13 | 0 | 4 |
| Ita | 99 | 99 | 99 | 99 | 50 | 55 | 37 | 13 | 18 | 23 | 8 | 3 |

Figure: Results show Ith (improvement over default-thread mapping) and Ita (improvement over default-task mapping) for the benchmarks.

# Result Analysis

In BenchErl/mbrot

- **WHERE**:
  - each Worker communicates with a ‘Mandel’ actor that checks if a pixel belongs to the Mandelbrot set or not,
- **PROBLEM**:
  - inefficient decision,
  - each **Worker** can perform this test independently,
  - introducing shared **Mandel** kills the parallel performance.
- **FIX**:
  - **Mandel** is assigned **MONITOR** execution policy,
  - each **Worker** now executes the **Mandel** actor’s code.

Can we reduce the performance penalties due to inefficient design of actor system?

**Future Work!!!**

# Limitations

- application of our technique to wide-variety of JVM-based actor frameworks, Call for collaborations!

# Limitations

- application of our technique to wide-variety of JVM-based actor frameworks, Call for collaborations!
- dynamism in actor-model
  - dynamic actor creation
    - execution policy for the actor type is still assigned!
  - dynamism in actor communication graph
    - our technique does not rely heavily on ACG, however availability of partial/full ACG helps to improve the mapping further!
    - also, programmers can use execution traces to gather ACG.

# Future Work

- **load-imbalance**,
  - assigning execution policy that enables **load-balancing**.

# Future Work

- **load-imbalance**,
  - assigning execution policy that enables **load-balancing**.
- **contentions** (bencherl/serialmsg)

| #cores | Th | Ta | H |
|---|---|---|---|
| 2 | 215097/36875 | 80912/23586 | 185018/3267 |
| 4 | 250674/21657 | 177178/25536 | 208674/1318 |
| 8 | 268856/5555 | 228561/9306 | 222940/1437 |
| 12 | 273426/5353 | 237860/25855 | 224680/1757 |

  - solution: ***contention-aware*** assignment of execution policy.

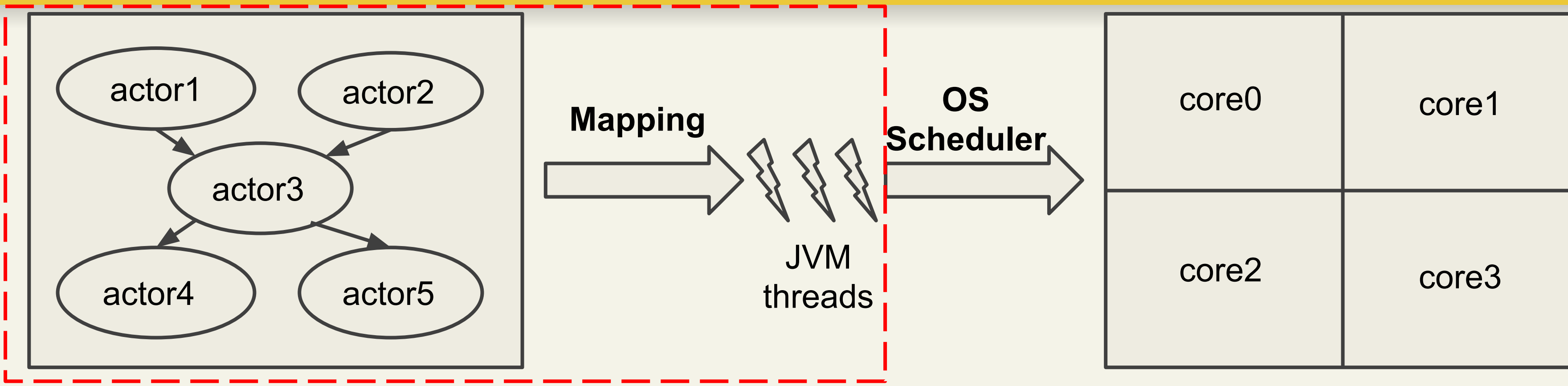
# Future Work

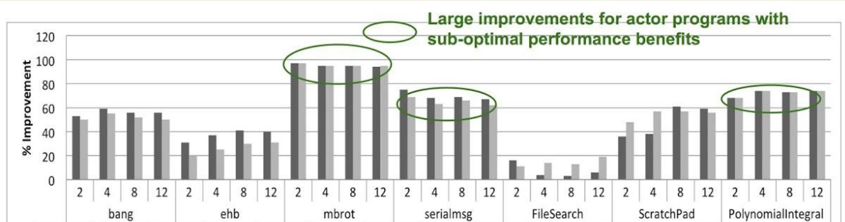
- **load-imbalance**,
  - assigning execution policy that enables **load-balancing**.
- **contentions** (bencherl/serialmsg)

| #cores | Th | Ta | H |
|---|---|---|---|
| 2 | 215097/36875 | 80912/23586 | 185018/3267 |
| 4 | 250674/21657 | 177178/25536 | 208674/1318 |
| 8 | 268856/5555 | 228561/9306 | 222940/1437 |
| 12 | 273426/5353 | 237860/25855 | 224680/1757 |

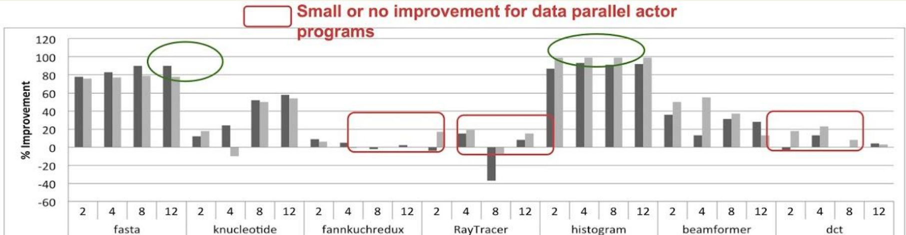
  - solution: ***contention-aware*** assignment of execution policy.
- **cache-miss** (FileSearch)
  - about 10% LLC-load-misses
  - solution: ***cache-aware*** assignment of execution policy.

# Conclusion




| | bang | | | | ehb | | | | mbrot | | | | serialmsg | | | | FileSearch | | | | ScratchPad | | | | PolynomialIntegral | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 53 | 59 | 56 | 56 | 31 | 37 | 41 | 40 | 97 | 95 | 95 | 94 | 75 | 68 | 69 | 67 | 16 | 4 | 3 | 6 | 36 | 38 | 61 | 59 | 68 | 74 | 73 | 74 |
| Ita | 50 | 55 | 52 | 50 | 20 | 25 | 30 | 31 | 97 | 95 | 95 | 95 | 69 | 63 | 66 | 62 | 11 | 14 | 13 | 19 | 48 | 57 | 57 | 56 | 68 | 74 | 73 | 74 |




| | fasta | | | | knucleotide | | | | fannkuchredux | | | | RayTracer | | | | histogram | | | | beamformer | | | | dct | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 | 2 | 4 | 8 | 12 |
| Ith | 78 | 83 | 90 | 90 | 12 | 24 | 52 | 58 | 9 | 5 | -2 | 2 | -4 | 15 | -37 | 8 | 87 | 93 | 91 | 92 | 36 | 13 | 31 | 28 | -4 | 13 | 0 | 4 |
| Ita | 76 | 77 | 79 | 78 | 18 | -10 | 50 | 54 | 6 | -1 | 0 | 0 | 17 | 19 | -8 | 15 | 99 | 99 | 99 | 99 | 50 | 55 | 37 | 13 | 18 | 23 | 8 | 3 |

# Questions?


Ganesha Upadhyaya
ganeshau@iastate.edu



This work was supported in part by the NSF grants CCF- 08-46059, CCF-11-17937, and CCF-14-23370.