

Process Cooperativity as a Feedback Metric in Concurrent Message-Passing Languages

Alexander Dean

Rochester Institute of Technology
Golisano College of Computer and Information Sciences

August 12, 2014

Process Cooperativity as a Feedback Metric

in Concurrent Message-Passing Languages

1 Background

- Runtime Scheduling
- Cooperativity
- Message Passing

2 ErLam Toolkit

- The Language
- Channel Implementations
- Simulation & Visualization

3 Scheduler Implementations

- Example Schedulers
- Feedback Mechanisms

4 Results

5 Conclusions & Future Work

- ErLam Toolkit
- Cooperative Schedulers
- Cooperativity as a Metric

Background

1 Background

- Runtime Scheduling
- Cooperativity
- Message Passing

Background: Runtime Scheduling

- Schedulers can be defined in a discrete manner:

- 1 *Choose* a process from set,
- 2 *Reduce* it,
- 3 *Update* private scheduler state.

Background: Runtime Scheduling

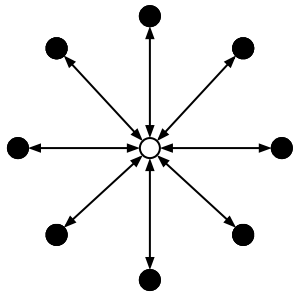
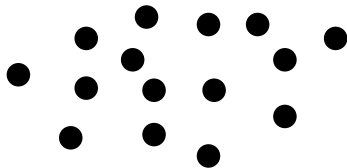
- Schedulers can be defined in a discrete manner:
 - 1 *Choose* a process from set,
 - 2 *Reduce* it,
 - 3 *Update* private scheduler state.
- Statistics can be gathered at every step about process:
 - Timestamp of last run,
 - Number of reductions, *etc.*

Background: Runtime Scheduling

- Schedulers can be defined in a discrete manner:
 - 1 *Choose* a process from set,
 - 2 *Reduce* it,
 - 3 *Update* private scheduler state.
- Statistics can be gathered at every step about process:
 - Timestamp of last run,
 - Number of reductions, *etc.*
- *What statistics are useful?*

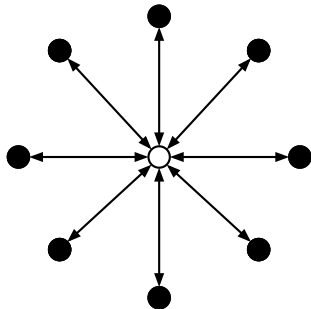
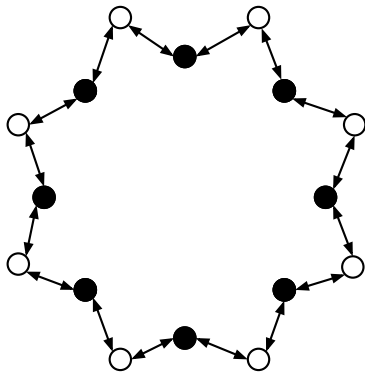
Background: Cooperativity

What is Process Cooperativity?



Background: Cooperativity

What does Cooperativity give us?



Background: Message Passing

We use a Symmetric, Synchronous, Message-Passing Primitive:

swap

- Purely captures cooperation of processes through synchronizing on a shared channel.

2 ErLam Toolkit

- The Language
- Channel Implementations
- Simulation & Visualization

ErLam Toolkit: The Language

```
<Expression> ::= <Variable>
                | <Integer>
                | 'newchan'
                | '(' <Expression> ')',
                | <Expression> <Expression>
                | 'if' <Expression> <Expression> <Expression>
                | 'swap' <Expression> <Expression>
                | 'spawn' <Expression>
                | 'fun' <Variable> '.' <Expression>
```

ErLam Toolkit: The Language

```
elib
  // ...
  ignore = (fun _.(fun y.y));
  omega = (fun x.(x x));
  // ...
  add = _erl[2]{ fun(X) when is_integer(X) ->
                  fun(Y) when is_integer(Y) ->
                      X+Y
                  end
                end
            };
  // ...
bile
```

ErLam Toolkit: The Language

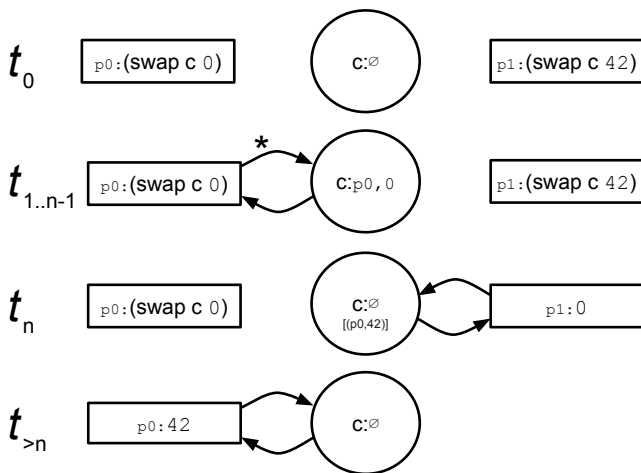
Example Application: Simple Swap

```
(fun c.  
  (ignore  
    (spawn (fun _.(swap c 42)))  
    (swap c 0))  
newchan)
```



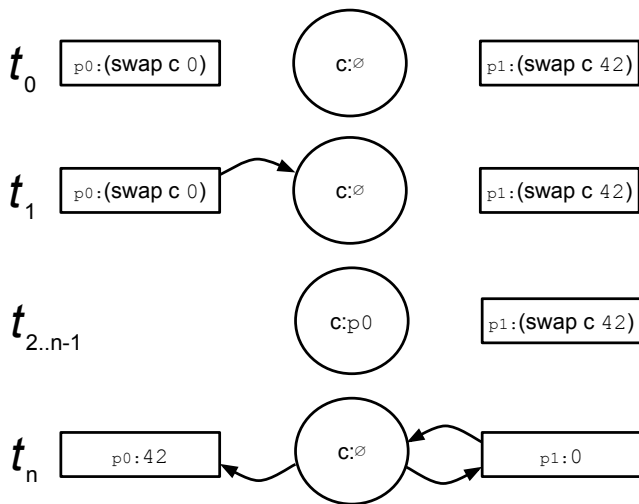
ErLam Toolkit: Channel Implementations

Process Blocking Swap



ErLam Toolkit: Channel Implementations

Process Absorption Swap



ErLam Toolkit: Simulation & Visualization

Primitive Testing Behaviours:

- Degree of Parallelism
- Partial System Cooperativity

Logging & Report Generation

ErLam Toolkit: Simulation & Visualization

System Behaviours: Degree of Parallelism

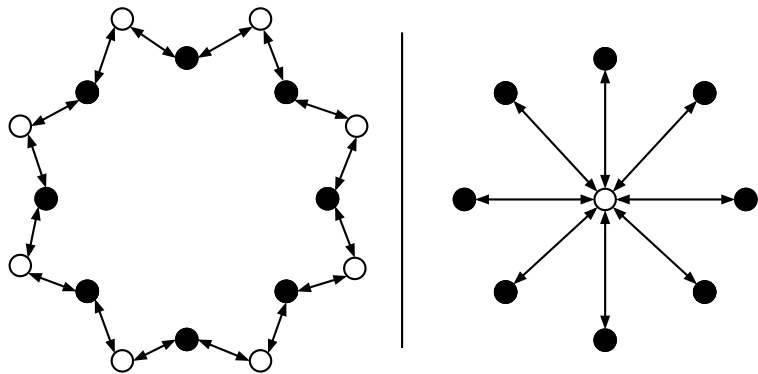


Figure: $PRing_N$, and $ClusterComm_{(N,1)}$ primitives to test degree of parallelism.

ErLam Toolkit: Simulation & Visualization

System Behaviours: Consistency of Cooperation

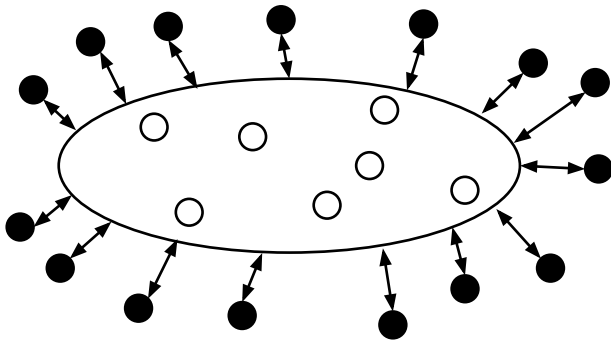


Figure: $ClusterComm_{(N,M)}$ to test effect of consistency on scheduler.

ErLam Toolkit: Simulation & Visualization

System Behaviours: Partial System Cooperativity

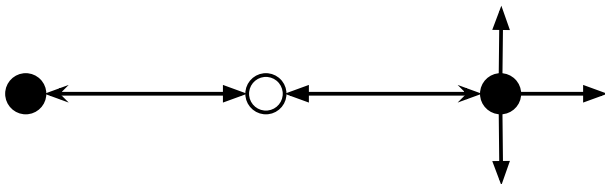


Figure: $UserInput_{(T,C)}$, simulates user interaction for a number (C) of external/timed (T) events.

ErLam Toolkit: Simulation & Visualization

System Behaviours: Partial System Cooperativity

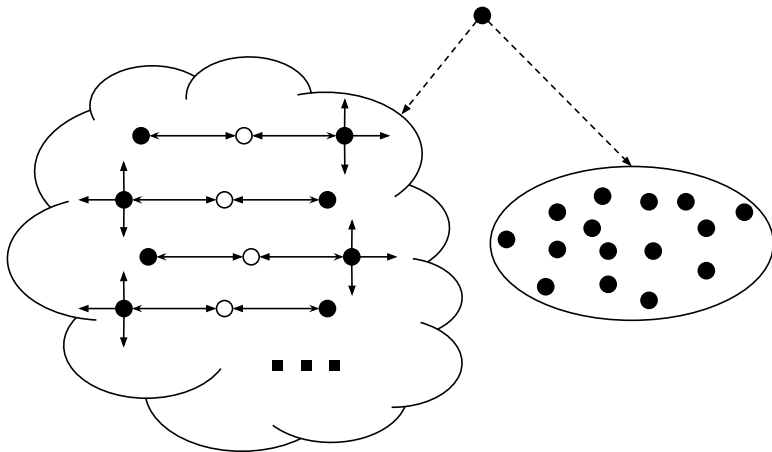


Figure: $Interactivity_{(N,M)}$, composure of $ChugMachine_N$ (Cloud), and M instances of $UserInput_{(5,2)}$.

ErLam Toolkit: Simulation & Visualization

System Behaviours: Partial System Cooperativity

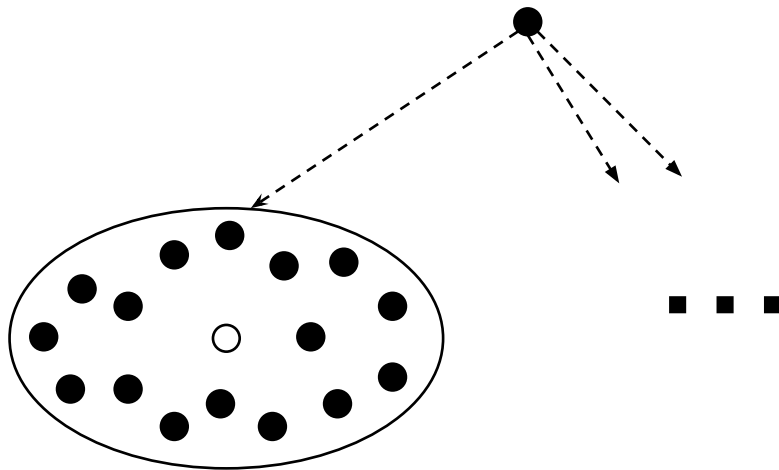


Figure: $PTree_{(W,N)}$, a composure of W $ClusterComm_{(N,1)}$ instances running concurrently.

ErLam Toolkit: Simulation & Visualization

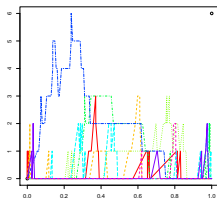
Logging & Report Generation

Things we could log:

- Process Queue Size (per LPU)
- Quantity of Reductions/Yields/Preempts
- State of the Scheduler (waiting/running)
- Channel State (Blocked/Unblocked)
- ...

ErLam Toolkit: Simulation & Visualization

Queue Size



Reduc. Density



Channel State



Scheduler Implementations

3 Scheduler Implementations

- Example Schedulers
- Feedback Mechanisms

Scheduler Implementations: Example Schedulers

- (MTRRGQ) - Round-Robin with Single Global Queue
 - All LPUs share a Process Queue.
- (MTRRWS-SQ) - Round-Robin with Work-Stealing via Direct Access
 - All LPUs have their own Process Queue.
 - LPUs can steal processes by grabbing them off the end of another LPU's queue.

Scheduler Implementations: Feedback Mechanisms

Three types of mechanics:

- Longevity-Based Batching
- Channel Pinning
- Bipartite-Graph Aided Sorting

Scheduler Implementations: Feedback Mechanisms

Longevity-Based Batching

- Choose via Round-Robin
 - from batch rather than queue
 - keeps track of number of rounds (batch size)
- Work-Steal whole batches
- Spawn to batch unless: $|b_i| \geq B$
 - Make singleton with new process.
 - Push parent and child into new batch.

Scheduler Implementations: Feedback Mechanisms

Longevity-Based Batching

- Choose via Round-Robin
 - from batch rather than queue
 - keeps track of number of rounds (batch size)
- Work-Steal whole batches
- Spawn to batch unless: $|b_i| \geq B$
 - Make singleton with new process.
 - Push parent and child into new batch.

GOAL: Can batching based on longevity account for fine/coarse parallelism in application?

Scheduler Implementations: Feedback Mechanisms

Channel-Pinning

- Upon call to *newchan*, pin to LPU based on spread algorithm:
 - *same* - LPU *newchan* is called is where it is pinned.
 - *even* - Cycle through LPUs and pin based on that.
 - ...
- Work-steal based on channel that's been pinned to you.

Scheduler Implementations: Feedback Mechanisms

Channel-Pinning

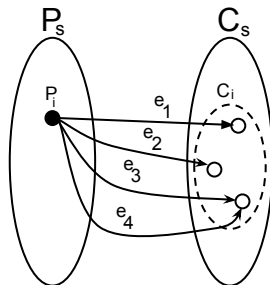
- Upon call to *newchan*, pin to LPU based on spread algorithm:
 - *same* - LPU *newchan* is called is where it is pinned.
 - *even* - Cycle through LPUs and pin based on that.
 - ...
- Work-steal based on channel that's been pinned to you.

GOAL: Can an *even*-like spread increase early saturation?

Scheduler Implementations: Feedback Mechanisms

Bipartite-Graph Aided Sorting

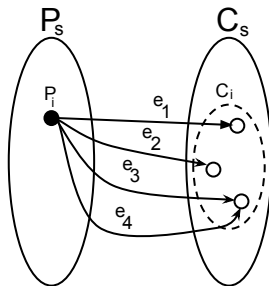
- Based on Round-Robin & Work-stealing
- Keep track of events which may effect cooperativity:
 - Spawning
 - Blocking/Unblocking
 - Steals
- If number of events over some threshold, re-sort.



Scheduler Implementations: Feedback Mechanisms

Bipartite-Graph Aided Sorting

- Based on Round-Robin & Work-stealing
- Keep track of events which may effect cooperativity:
 - Spawning
 - Blocking/Unblocking
 - Steals
- If number of events over some threshold, re-sort.



GOAL: Are alternate channel implementations worth exploration?

| 4 Results

Results:

Longevity-Based Batching

- *Can batching based on longevity recognize fine/coarse parallelism in an application?*

Channel Pinning

- *Can an even-like spread increase early saturation?*

Bipartite-Graph Aided Sorting

- *Are alternate channel implementations worth exploration?*

Results: Longevity-Based Batching

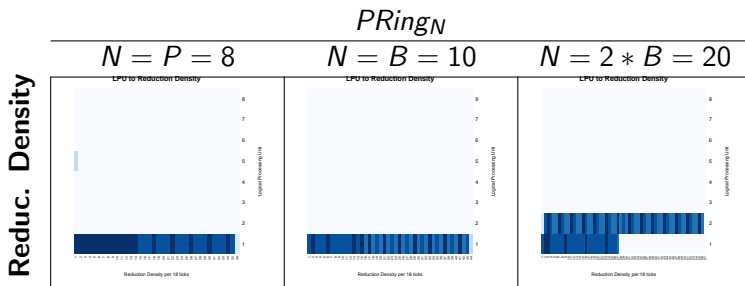
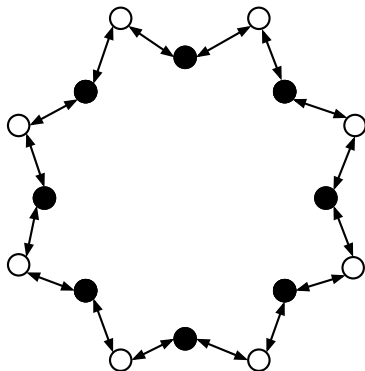
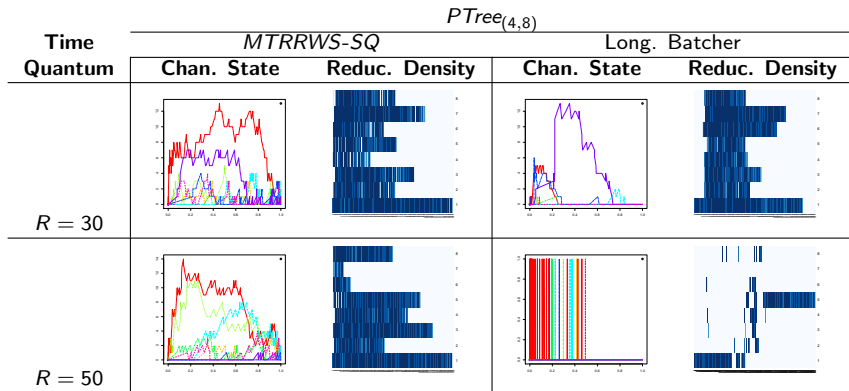


Table: Comparison of different sized $PRing_N$ on the Longevity Batching Scheduler with batch size $B = 10$.

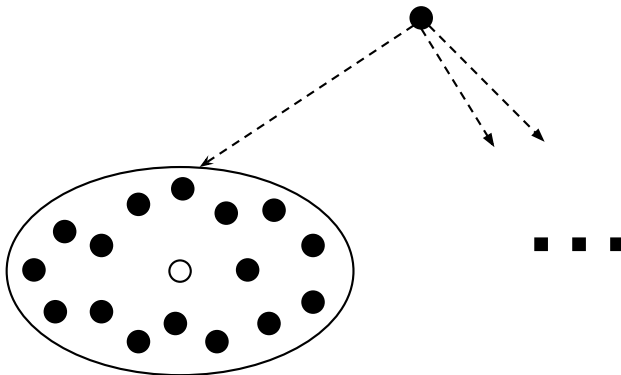
Results:



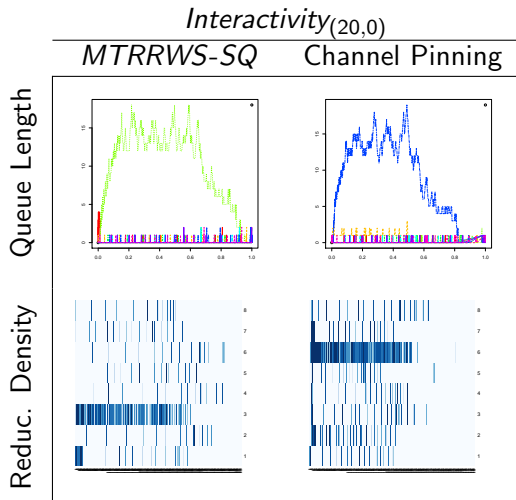
Results: Longevity-Based Batching



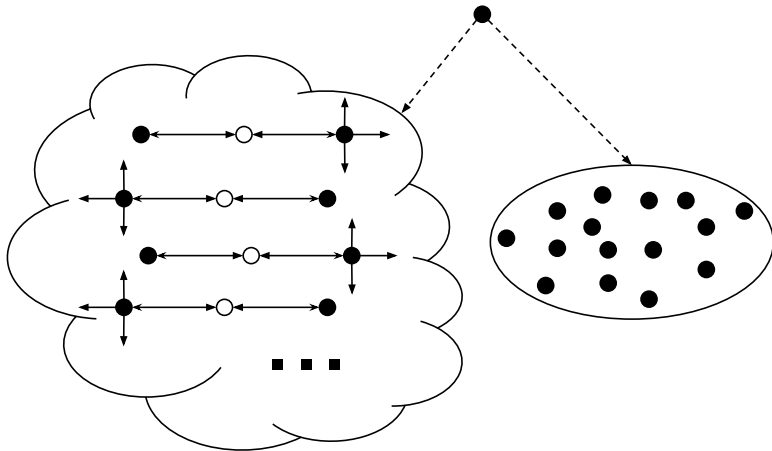
Results:



Results: Channel Pinning

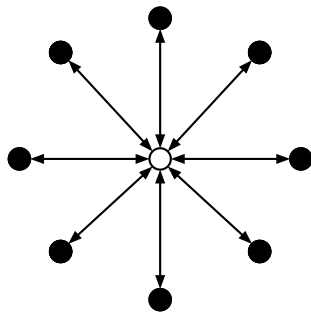
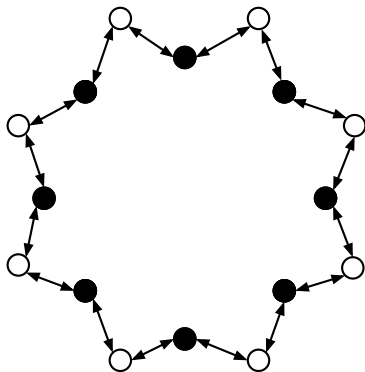


Results:



Results:

Results: Bipartite-Aided Graph Sorting



Results:

Results: Bipartite-Aided Graph Sorting

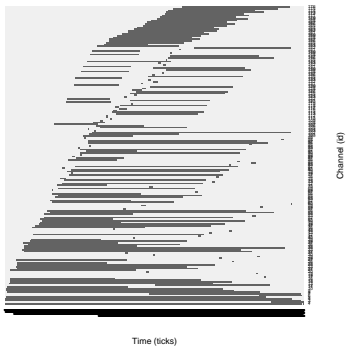
```
fun N.  
  (omega fun f,m.(  
    if (leq m 1)  
      m  
      (merge fun _.(f f (sub m 1))  
              fun _.(f f (sub m 2))  
              add)) N)
```

Results: Bipartite-Aided Graph Sorting

Parallel Fibonacci

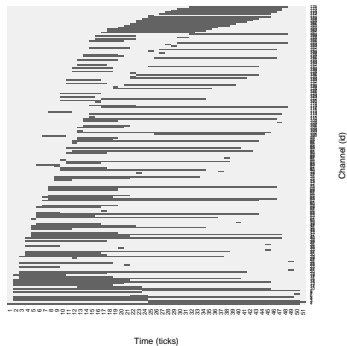
MTRRWS-SQ

Channel State Over Time



Sorting Scheduler

Channel State Over Time



Conclusions & Future Work

5 Conclusions & Future Work

- ErLam Toolkit
- Cooperative Schedulers
- Cooperativity as a Metric

Conclusions & Future Work: ErLam Toolkit

- Test Primitives were nicely composable process behaviours.
 - More research into generating behaviours.
 - More compositions: PTree with Rings.
- Log generation, lots of overhead, but good observations.

Conclusions & Future Work: Cooperative Schedulers

■ Longevity Batching:

- Would benefit from heuristic based Quantum selection.
- As it stands, limited gain from longevity recognition.

■ Channel Pinning:

- Promising saturation and work-stealing mechanic.

■ Bipartite-Graph Aided Sorting:

- Supprising results on MapReduce style applications.
- Worth studying Blocking-Channels further for gains from sorting.

Conclusions & Future Work: Cooperativity as a Metric

- Possible to recognize and benefit from.
- The three example mechanics are promising and can be extended for practice modern languages, despite simplistic simulation language.
- More to explore:
 - Alternate Message-Passing Types (Asymmetric?)
 - ...

Questions/Comments?

Questions/Comments?

Thank You!

Links

- <https://github.com/dstar4138/erlam>
- https://github.com/dstar4138/thesis_cooperativity
- <http://dstar4138.com>