Towards Pre-Deployment Detection of Performance Failures in Cloud Systems

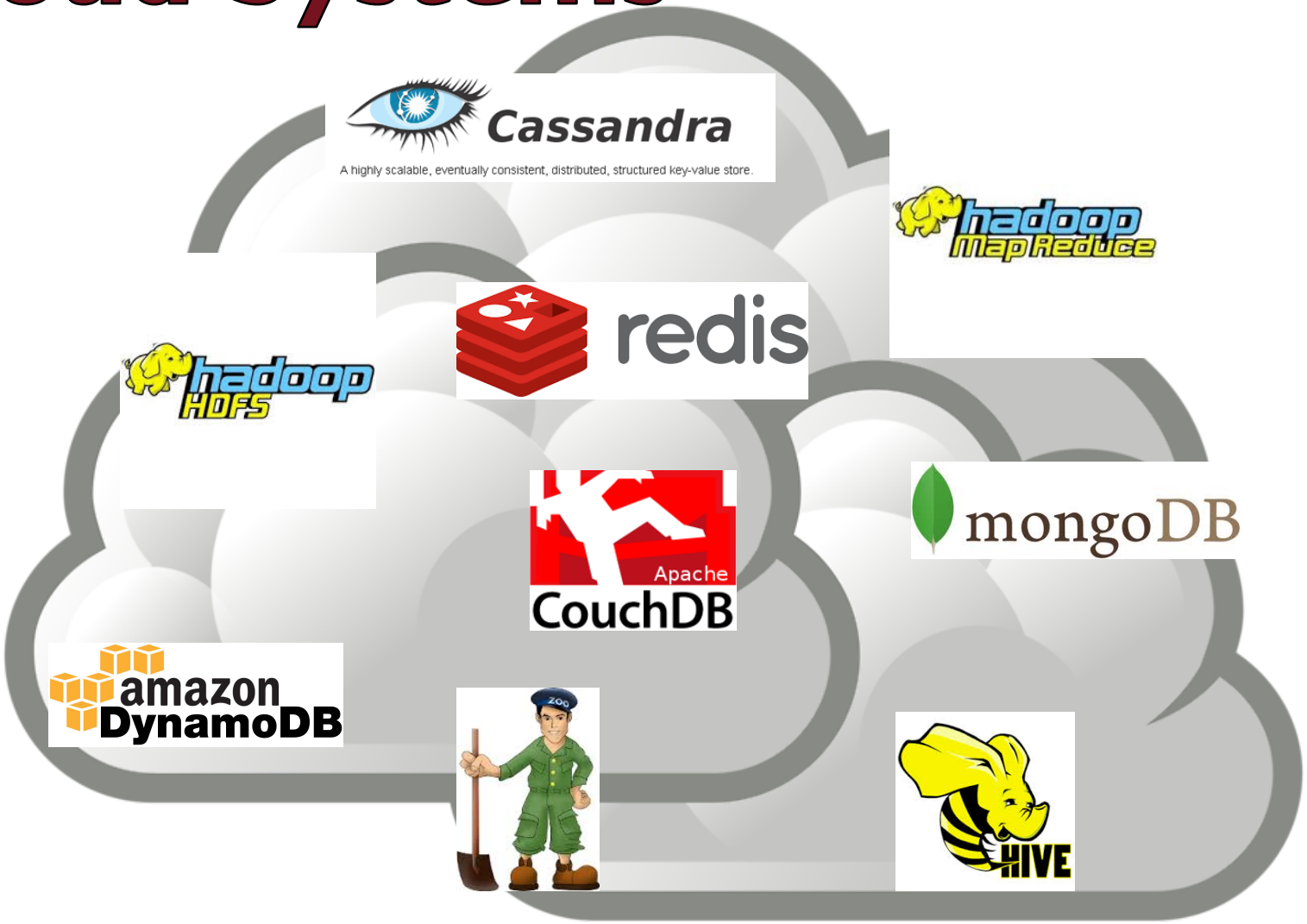
Riza Suminto, Agung Laksono*, Anang Satria*,
Thanh Do†, Haryadi Gunawi

* SURYA UNIVERSITY

THE UNIVERSITY OF CHICAGO †

Microsoft GRAY SYSTEMS LAB

Cloud Systems

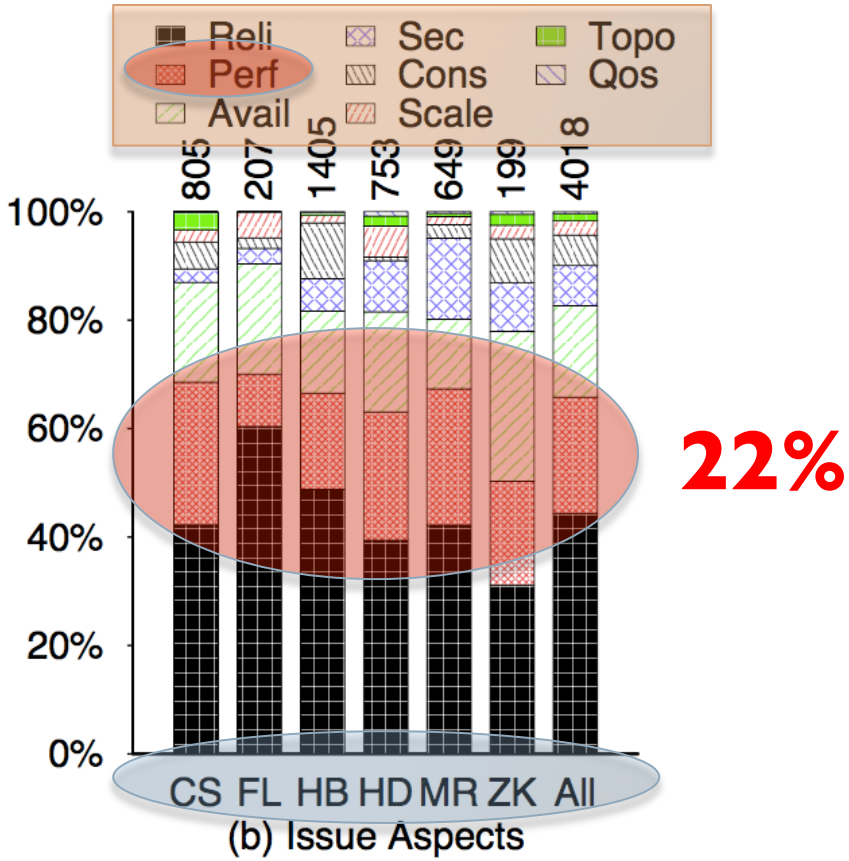


Demands

- Users demand high dependability, reliability, and performance stability
- Amazon found that every 100ms of latency cost them 1% in sales
- Google found an extra 0.5 second in search page generation time dropped traffic by 20%

Speed Matters!

Performance failures happen



22%

What Bugs Live in the Cloud? A Study of 3000+ Issues in Cloud Systems, SOCC'14

Outline

Performance Bug



System Performance Verifier

Performance Bug

- Jobs take multiple times than usual to finish
  - Improper speculative execution

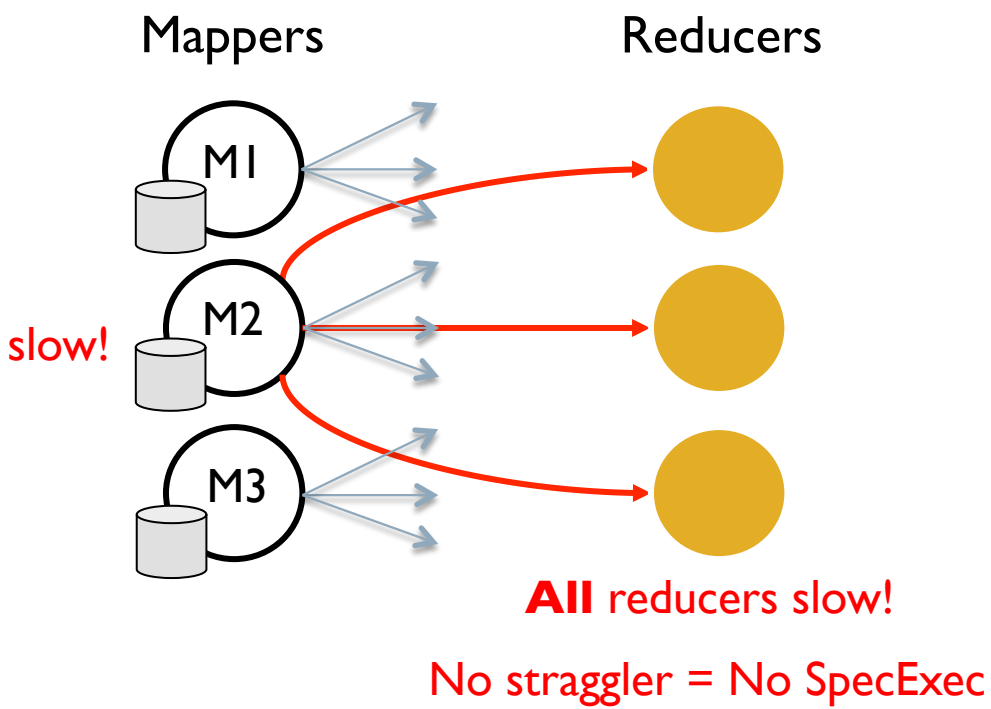
    JCH_1 & TPL_1 & FPL_2 & FTY_1
  - Unnecessary repeated recovery

    TPL_1 & TPL_4 & FTY_4 & TOP_1

Scenario Type	Possible Conditions
DLC: Data Locality	(1) Read from remote disk, (2) read from local disk, ...
DSR: Data Source	(1) Some tasks read from same datanode, (2) all tasks read from different datanodes, ...
JCH: Job Characteristic	Map-reduce is (1) many-to-all, (2) all-to-many, (3) large fan-in, (4) large fan-out, ...
JSZ: Job Size	(1) 1 GB jar file, (2) 1 MB jar file, ...
LSZ: Load Size	(1) Thousands of tasks, (2) small number of tasks, ...
FTY: Fault Type	(1) Slow node/NIC, (2) Node disconnect/packet drop, (3) Disk error/out of space, (4) Rack switch, ...
FPL: Fault Placement	Slowdown fault injection at the (1) source datanode, (2) mapper, (3) reducer, ...
FGR: Fault Granularity	(1) Single disk/NIC, (2) single node (deadnode), (3) entire rack (network switch), ...
FTM: Fault Timing	(1) During shuffling, (2) during 95% of task completion, ...
TOP: Topology Scenario	(1) 30 nodes per rack, (2) 3 nodes per rack, ...
TPL: Task Placement	(1) Mappers and reducers are in different nodes, (2) AM and reducers in different nodes, (3) Mappers are in the same node, (4) Most of reducers placed in the same rack, ...

Untriggered SpecExec

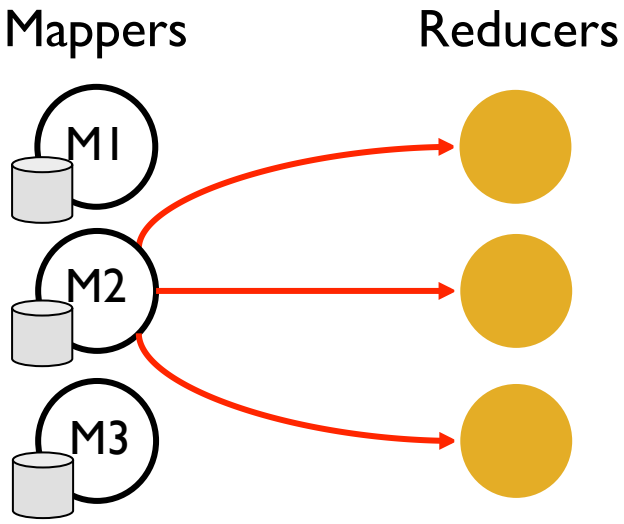
$\mathbf{DLC_A}$	Map read locally
$\mathbf{TPL_A}$	Mappers and reducers in different nodes
$\mathbf{JCH_A}$	All-to-All
$\mathbf{FPL_A}$	Fault at map node
$\mathbf{FTY_A}$	Slow NIC



$$DLC_A \,\&\, TPL_A \,\&\, JCH_A \,\&\, FPL_A \,\&\, FTY_A$$

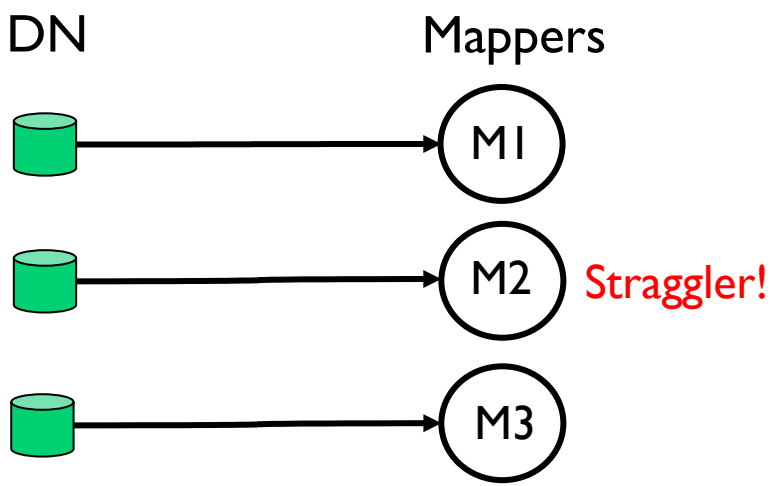
Untriggered SpecExec, cont

- DLC_A & TPL_A & JCH_A & FPL_A & FTY_A



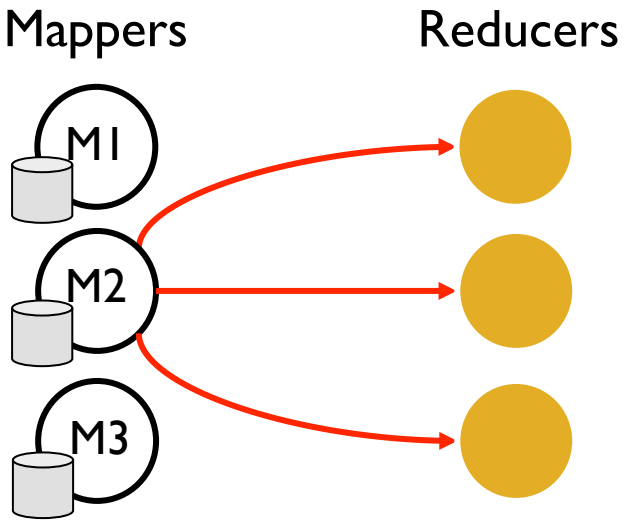
- ~~DLC_A~~ & TPL_A & JCH_A & FPL_A & FTY_A

DLC_B = read remote



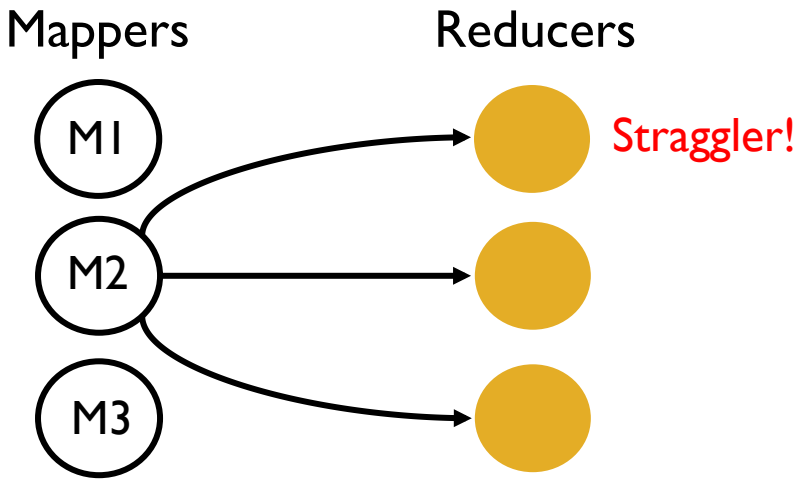
Untriggered SpecExec, cont

- DLC_A & TPL_A & JCH_A & FPL_A & FTY_A



- DLC_A & TPL_A & JCH_A & ~~FPL_A~~ & FTY_A

slow reducer = FPL_B



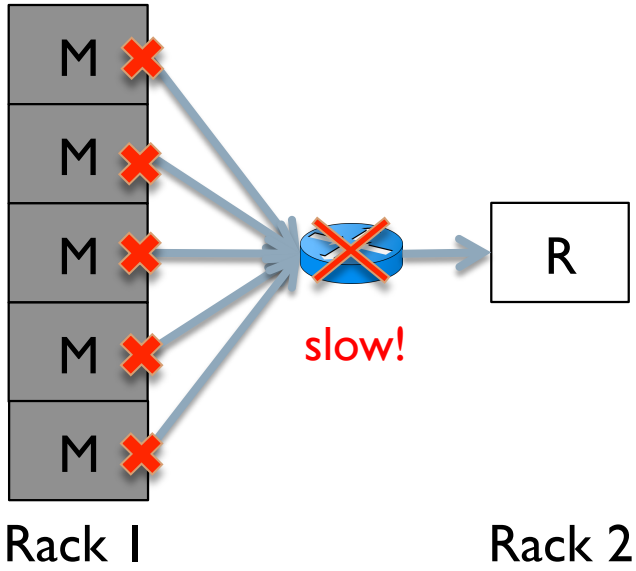
O(n) Recovery

TPL_A Mappers and Reducers in different nodes

TPL_B Mappers and Reducers in different racks

TOP_A Large number of nodes per rack

FTY_B Slow inter-rack switch



TPL_A & TPL_B & TOP_A & FTY_B

Conditions lead to performance bug

Scenario Type	Possible Condition
DLC: Data Locality	(1) Read from remote disk, (2) read from local disk, ...
DSR: Data Source	(1) Some tasks read from same datanode, (2) all tasks read from different datanodes, …
JCH: Job Characteristic	Map-reduce is (1) many-to-all, (2) all-to-many, (3) large fan-in, (4) large fan-out, ...
JSZ: Job Size	(1)1GBjarfile,(2)1MBjarfile,...
LSZ: Load Size	(1) Thousands of tasks, (2) small number of tasks, …
FTY: Fault Type	(1) Slow node/NIC, (2) Node disconnect/packet drop, (3) Disk error/out of space, (4) Rack switch, …
FPL: Fault Placement	Slowdown fault injection at the (1) source datanode, (2) mapper, (3) reducer, …
FGR: Fault Ganularity	(1) Single disk/NIC, (2) single node (deadnode), (3) en- tire rack (network switch), …
FTM: Fault Timing	(1) During shuffling, (2) during 95% of task completion, …
TOP: Topology	(1) 30 nodes per rack, (2) 3 nodes per rack, …
TPL: Task Placement	(1) Mappers and reducers are in different nodes, (2) AM and reducers in different nodes, (3) Mappers are in the same node, (4) Most of reducers placed in the same rack, ...

- Untriggered Speculative Execution
  - MR-70001 = JCH_1 & TPL_1 & FPL_2 & FTY_1
  - MR-70002 = DSR_1 & DLC_1 & FPL_1 & FTY_1
  - MR-5533 = FTY_2 & FPL_3 & TPL_3
  - …
- O(n) Recovery
  - MR-5251 = FTY_3 & FPL_3 & FTM_1
  - MR-5060 = TPL_1 & TPL_3 & FTY_1 & FPL_2
  - MR-1800 = TPL_1 & TPL_4 & FTY_4 & TOP_1
  - …
- Long lock contention
  - MR-9191 = FTY_3 & FPL_3 & FTM_1
  - MR-9292 = TPL_1 & TPL_3 & FTY_1 & FPL_2
  - MR-9393 = TPL_1 & TPL_4 & FTY_4 & TOP_1
  - …

Outline

Performance Bug



System Performance Verifier

Current Approach

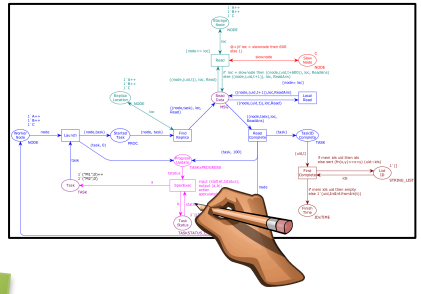
- Benchmarking
- **Hundreds** benchmark for every scenario
- Injecting slowdowns and failures
- Take **days to weeks!!**

What we want...

- Four goals in performance verification
  - Fast
  - Covers many deployment scenario
  - Runs in pre-deployment
  - Directly checks implementation code

Formal modeling tools!

System Performance Verifier (SPV)

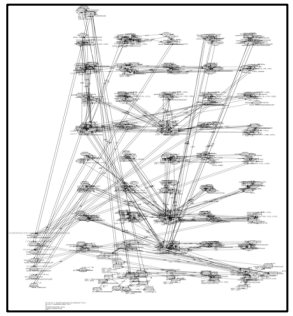


Hand model

20X larger than hand model

```
public synchronized void launchTask(RunningJob rjob) throws IOException {
  @Data
  public class JobInProgress {
    JobID jobId;
    TaskInProgress maps[];
    ...
  }
  @IO
  public HeartbeatResponse heartbeat
    (HeartbeatData hd){
    ...
  }
```

SPV Compiler



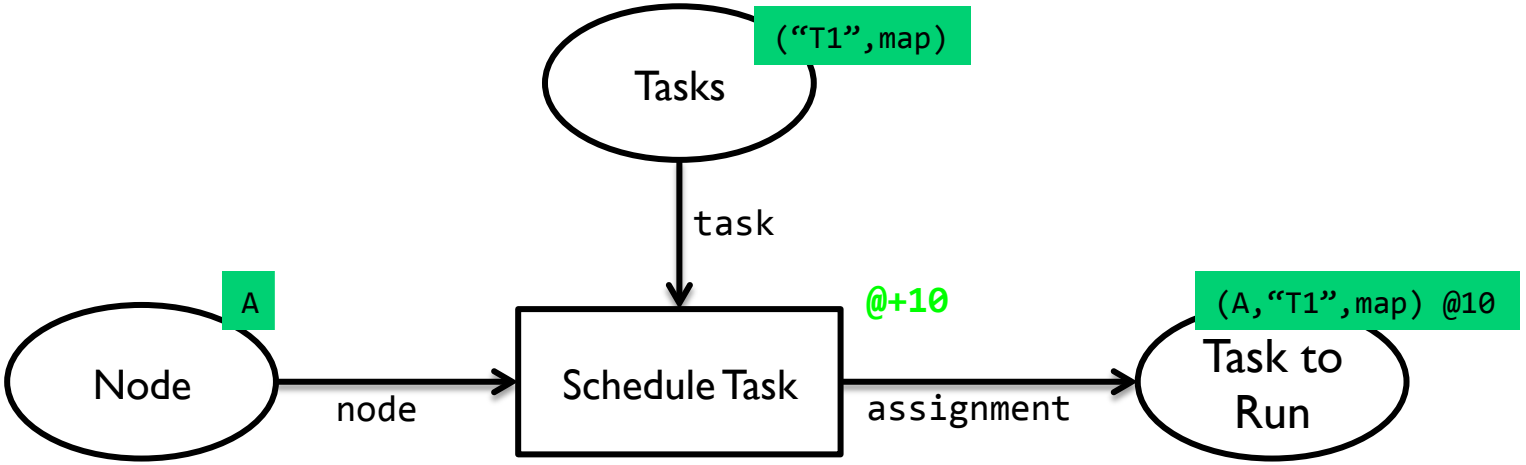


Target system
(e.g., Hadoop code)

Auto-generated model
(in Colored Petri Net)

Performance Verification

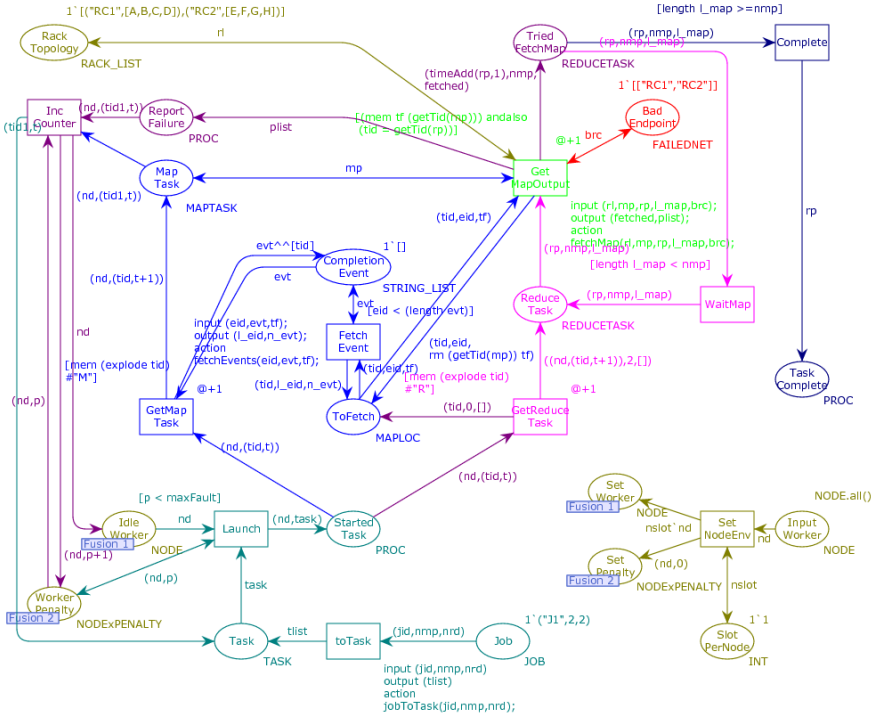
Colored Petri Nets (CPN)




```
input(node,task);
output(assignment);
action
  let val (id,type) = task
in
  (node,id,type)
end;
```

Challenges : Two Different World

CPN



Java

```
public synchronized void launchTask(RunningJob rjob) throws IOException {
  if (this.taskStatus.getRunState() == TaskStatus.State.UNASSIGNED ||
      this.taskStatus.getRunState() == TaskStatus.State.FAILED_UNCLEAN ||
      this.taskStatus.getRunState() == TaskStatus.State.KILLED_UNCLEAN) {
    localizeTask(task);
    if (this.taskStatus.getRunState() == TaskStatus.State.UNASSIGNED) {
      this.taskStatus.setRunState(TaskStatus.State.RUNNING);
    }
    setTaskRunner(task.createRunner(TaskTracker.this, this, rjob));
    this.runner.start();

    long now = System.currentTimeMillis();
    this.taskStatus.setStartTime(now);
    this.lastProgressReport = now;
  } else {
    LOG.info("Not launching task: " + task.getTaskID() +
        " since it's state is " + this.taskStatus.getRunState());
  }
}
```

Our Approach

- Java → SysJava
  - Data flattening
  - Code modularization
  - Annotation tagging
- SysJava → Model compiler

Data Flattening

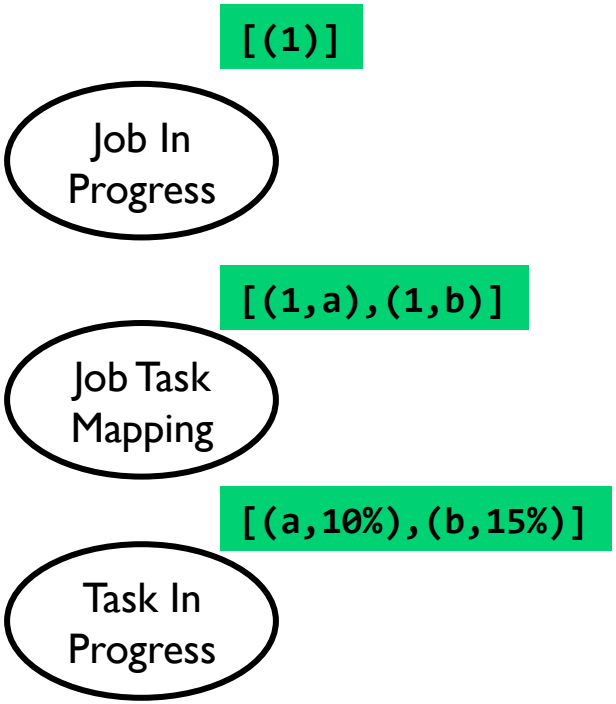
- Java system states = ArrayList, Map, Tree, . . .
- CPN states = multisets

```
List<JobInProgress> runningJobs;

public class JobInProgress {
  JobID jobId;
  TaskInProgress maps[];
  ...
}

class TaskInProgress {
  TaskID id;
  double progress;
  ...
}
```





Code Modularization

```
private boolean processHeartbeat(
  TaskTrackerStatus trackerStats) {

  synchronized (taskTrackers) {
    ...
  }

  for (TaskStatus ts: trackerStats) {
     tasks.get(ts.id).updateStatus(ts);
  }

  ...
}
```



Modular function

```
@ProcessState
private void initCheck() {
  synchronized (taskTrackers) {
    ...
  }
}
```

Control Flow logic

```
@ForEach
private void updateStatuses(
  TaskTrackerStatus trackerStats) {
  for (TaskStatus ts: trackerStats) {
    ...
  }
}
```

CRUD Logic

```
@GetState
private TaskInProgress getTask(TaskID id) {
  tasks.get(ts.id);
}
@UpdateState
private void tipUpdate(TaskInProgress tip,
  TaskStatus ts) {
  tip.updateStatus(ts);
}
```

Annotation Tagging

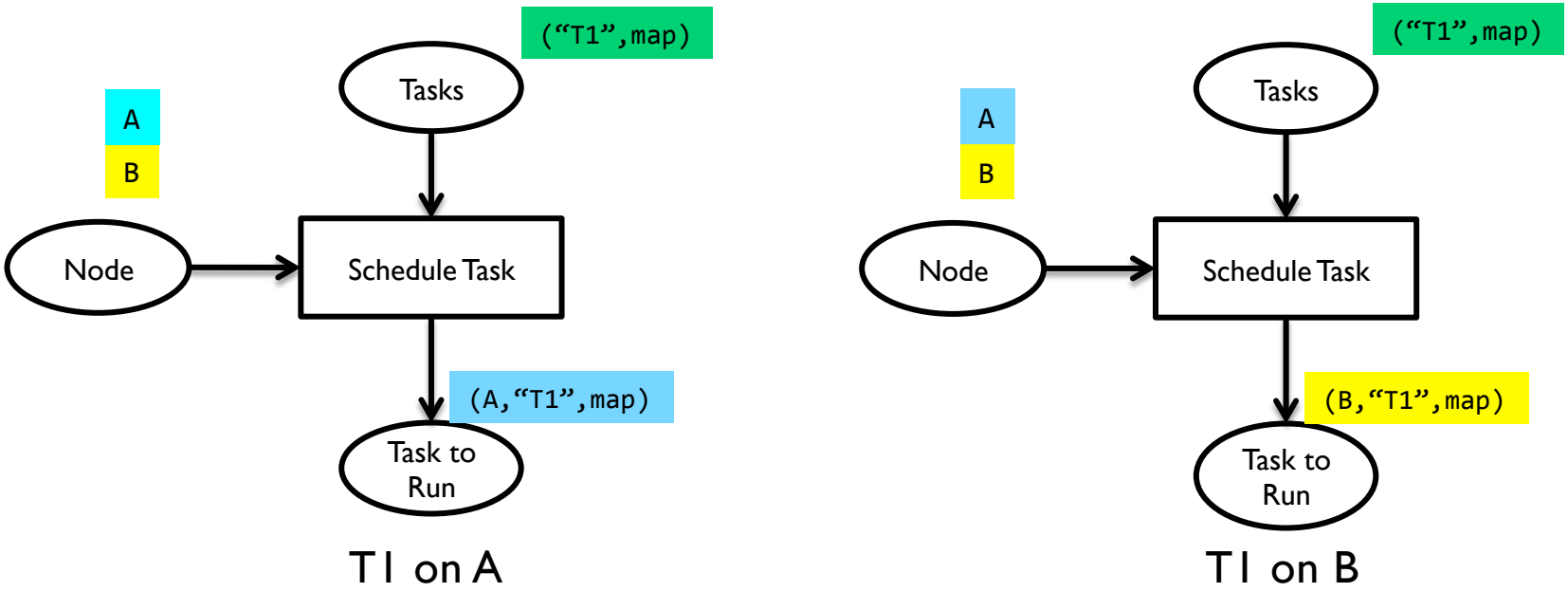
- Assist compiler
- Annotation Category:
  - Data Structure
  - I/O
  - CRUD & Process
  - Miscellaneous

```
@Data
public class JobInProgress {
  JobID jobId;
  TaskInProgress maps[];
  ...
}
```

```
@IO
public HeartbeatResponse heartbeat
  (HeartbeatData hd) {
  ...
}
```

Model Checking

- SPV Compiler ➔ Executable XML
- Define configurations, assertions, and specifications
- Explore every non-deterministic choices
  - Task to node mapping



Preliminary Result

- **5305** lines of code on top of WALA & Access/CPN
- Hadoop MapReduce 1.2.1, with **1067** lines code change
- **20x** larger than hand-made model
- **34** scenario, **30** assertion violation, **4** performance bug
- **1.5** hour model checking

Configuration	Value
Worker Node	Node A, B
Data Node	Node A, B, C
Tasks	2 Task
Fault Type	Slow Data Node
Fault Placement	Node C

Thank you!
Questions?





http://ucare.cs.uchicago.edu

Discussion

- Is it time for pre-deployment detection of performance bugs?
- Bridging system code and formal methods
- Future of data-centric languages
- Beyond Hadoop
- Root cause anatomy of performance bugs
- Beyond performance bugs