Fault Tolerance for Synchronous Streaming Programs

Nic Hollingum
Andrew Santosa
Bernhard Scholz

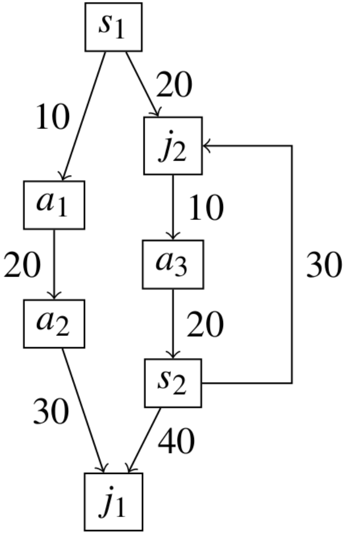
USYD

2012

Nic Hollingum
Andrew Santosa
Bernhard Scholz

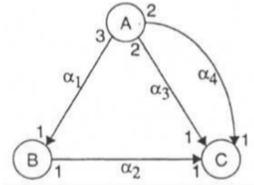
Dataflow

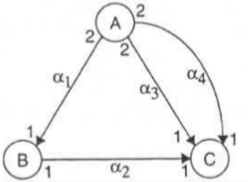
- Computational model [3]
  - No (or limited) main-memory
  - Communicating processes
  - Data filters
- Also hardware implementations [2]



Actors, Channels & Tokens

- Data sent around network as *tokens*
- Computational units, *actors*, process tokens and output new ones
- Tokens sent between actors via FIFO buffers called *channels*
- All token production rates known statically
- Compute *Steady State Schedule*, delays have no change
- Schedule *synchronises* actor executions

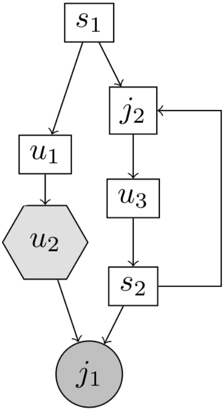


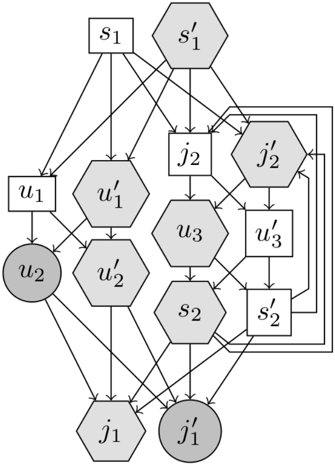


Processor Faults

- Running SDF programs on real computers
- One node failure every 100 hours [4]
- MapReduce [1]
  - Well known cloud service
  - Explicit fault-tolerance mechanisms
  - Survives worker faults, not master faults
- Fault tolerance necessary to make stream paradigm available as a service

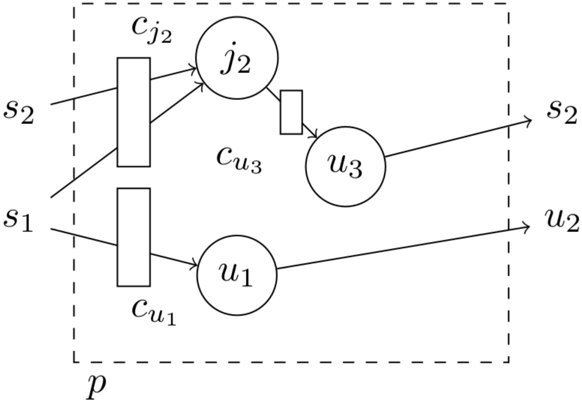
Replication



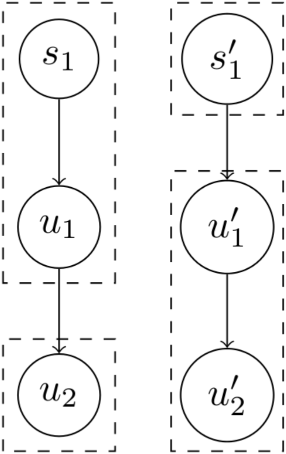


Nic Hollingum
Andrew Santosa
Bernhard Scholz

Checkpointing

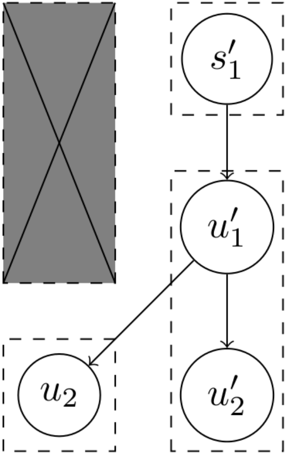


Dynamic-Switching



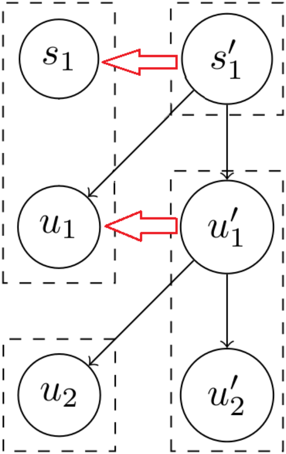
- Hybrid
  - Replicate: two distinct graphs
  - Checkpoint: in-memory state history
- No communication between graph sides
- Actors have *partners*
- Synchronise to prevent unbounded memory

Dynamic-Switching, Failure



- TCP timeout
- Parents stop sending data
- Partners buffer tokens
- Token Requests

Dynamic-Switching, Recovery

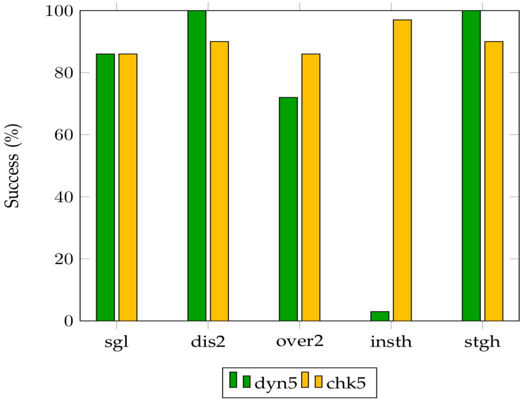


- Re-connect lost channels
- Partner recovery protocol
  - send missing tokens
  - send channel configuration
  - adopt partner's state
  - block during recovery
- Available for reconnection

Experimentation

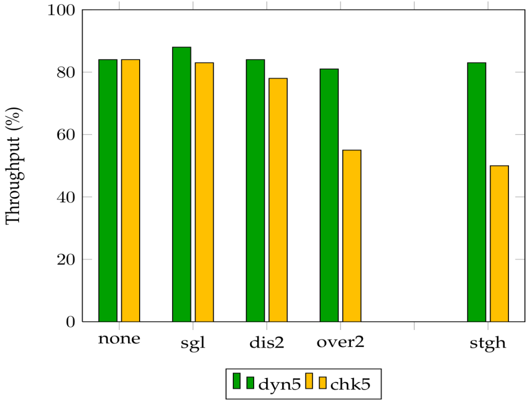
- Faults
  - Single, Distinct, Overlapping
  - Instant-half, Staggered-half
- Software
  - Java Open JDK 1.6
  - TCP/IP socket implementation
  - LAN configurable
  - StreamIt [5] benchmarks
  - Simulator
- Hardware
  - 20 low-end computers
  - Core2 duo E8400 2x3.0Ghz, 4GB RAM
  - Gigabit Ethernet

Resilience



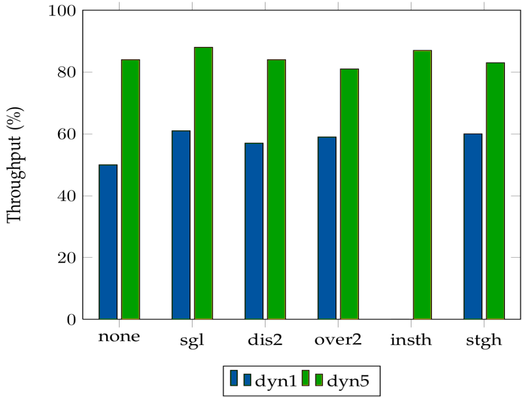
- Successful completions, dynamic vs. checkpointing
- Overlapping faults show difference

Overhead



- Throughput, dynamic vs. checkpointing
- Dynamic has minimal falloff

Synchrony



- Overheads - Memory bounding
- Wasting time blocking for catchup

Conclusion

- SDF paradigm suited to HPC
- Exploit unique properties of SDF for fault tolerance
- Develop distributed algorithms providing fault tolerance
- Examine on real hardware

References I

J. Dean and S. Ghemawat.
MapReduce: Simplified data processing on large clusters.
Comm. ACM, 51(1):107–113, 2008.

J.R. Gurd, C.C. Kirkham, and I. Watson.
The manchester prototype dataflow computer.
Comm. ACM, 28(1):34–52, 1985.

Richard M. Karp and Raymond E. Miller.
Properties of a model for parallel computations: Determinancy, termination, queueing.
SIAM Journal on Applied Mathematics, 14(6):1390–1411, 1966.

References II

Daniel A. Reed, Charng da Lu, and Celso L. Mendes.
Reliability challenges in large systems.
Future Generation Computer Systems, 22(3):293–302, 2006.

W. Thies and S. Amarasinghe.
An empirical characterization of stream programs and its implications for language and compiler design.
In *19th PACT*, pages 365–376. ACM, 2010.

Contributions

- Fault Tolerant Algorithms
  - Replication
  - Checkpointing
  - Dynamic Switching
- Experimental analysis
  - Checkpointing more resilient when faults overlap
  - Dynamic Switching more consistent throughput
  - Make throughput / memory-footprint tradeoff