

Advanced Synchronization Facility

Introducing ASF 2.0

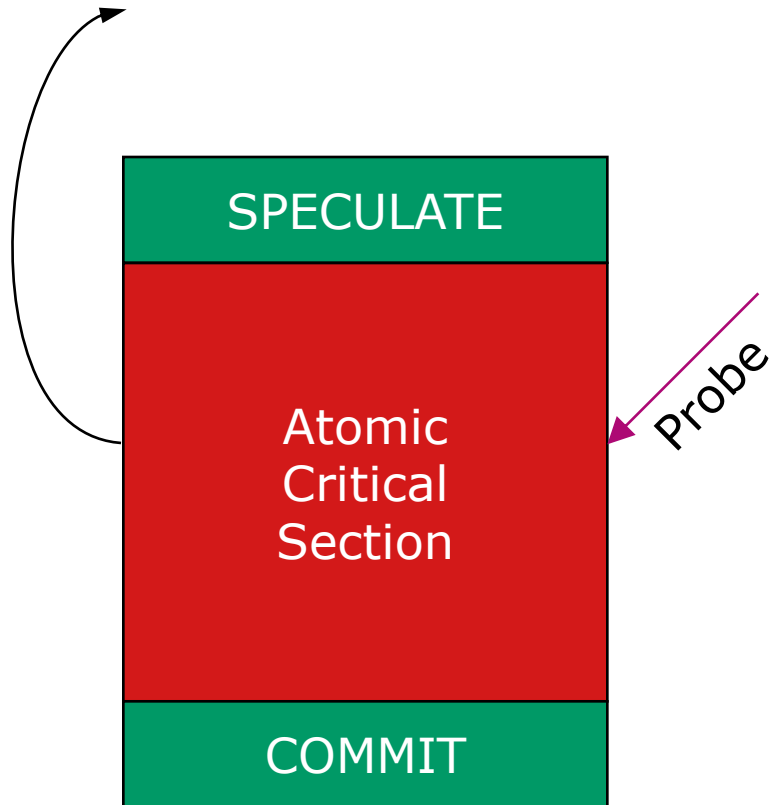
Stephan Diestelhorst | October 17, 2008



Functionality



ASF 2.0 – Overview



Atomic RMW construct

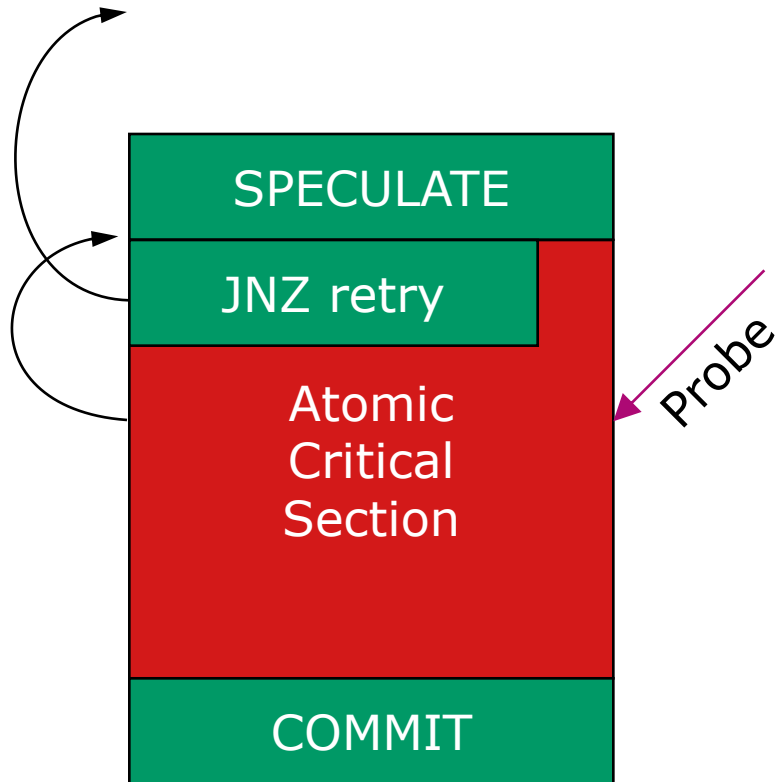
- Flexible
- Multiple locations
- Normal instruction set

General Idea

- Speculative execution
- Monitor for inconsistent probes
- Undo modifications & redirect control flow



ASF 2.0 – Primitives



SPECULATE

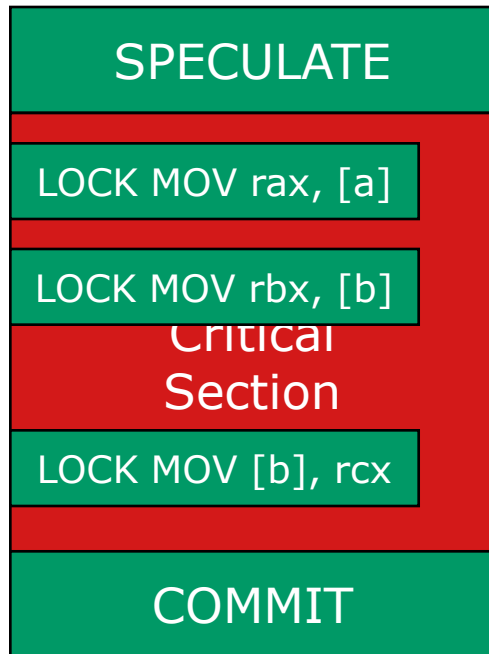
- Start critical section
- Anchor for later roll-back due to inconsistent probe
 - Keep: rIP, rSP
 - Don't keep: GPR, XMM, MMX, FPU,...
- Roll-back
 - Jump behind SPECULATE
 - Non-zero "return" value

COMMIT

- Stop monitoring
- Commit updates



ASF 2.0 – Primitives

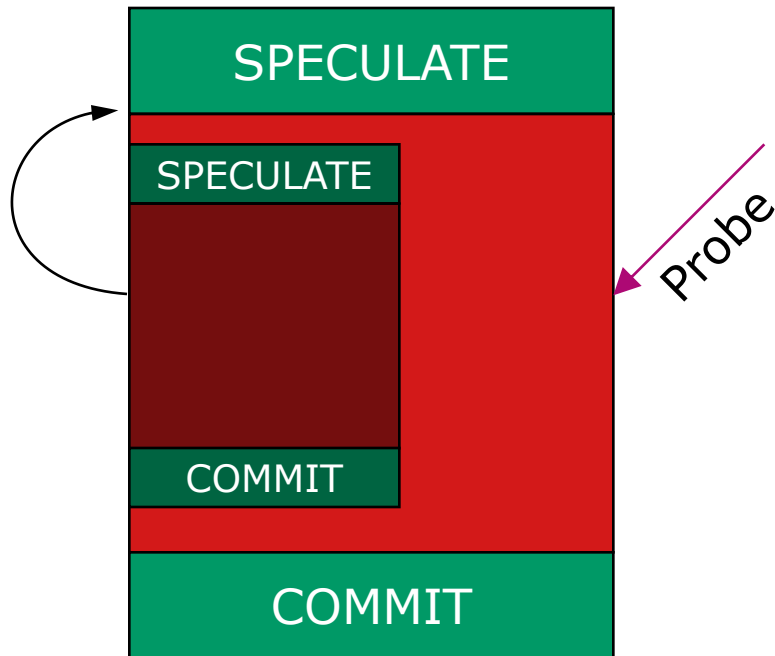


LOCK load / store

- Prefixed loads (mov..., prefetch, prefetchw)
 - Monitor these **cache lines** for probes
 - “Declarators”
- Prefixed stores (mov...)
 - Modify monitored (aka. protected) lines
 - Stores subject to undo / roll-back



ASF 2.0 – Nesting



- ASF supports flat nesting
 - Outer and inner lines must fit into capacity
 - Roll-back to the outermost SPECULATE
- Maintain internal nest count
 - SPECULATE increments
 - COMMIT decrements
- Inner SPECULATE & COMMIT become no-ops (except for inc / dec)



ASF 2.0 – Progress & Peculiarities

Progress

- Concurrent crit. sections can constantly abort each other and restart
- System-wide livelock possible -> back-off strategies

Peculiarities

- ASF's state not architectural
 - Roll-back on far-control transfer
 - “Imprecise”-bit in rFLAGS stack image-> true rIP in MSR
 - SVM detection: #GP on interceptable instructions



ASF 2.0 – Implementation

Various possible implementations

- Backup storage for original, unmodified lines
 - Allows cache evictions (must still see probes)
 - Decoupled from caches
- Use caches and store buffer
- ...

Size

- Minimal guaranteed capacity, implementation dependent
 - At least 4 cache lines, for all implementations
 - CPUID



Use Cases



Use Cases – Plain ASF

Flexible atomic primitives

- Multi-word CAS
- LL / SC equivalent constructs


Lock-free datastructures

Optimistic lock observation



Use Cases – Plain ASF

DCAS:

	SPECULATE	
	MOV RCX, 0	; Critical section begins
	JNZ fail	; Bail out if rolled back
	LOCK MOV R8, [mem1]	; Specification begins
	LOCK MOV R9, [mem2]	
	CMP R8, RAX	
	JNZ out	
	CMP R9, RBX	
	JNZ out	
	LOCK MOV [mem1], RDI	; Update protected memory
	LOCK MOV [mem2], RSI	
	MOV RCX, 1	
	out:	
	COMMIT	; End of critical section
	...	
	fail: ...	



Use Cases – Accelerating STM

Software Transactional Memory

- Transactions in main memory
- Atomic, tracks accessed locations with meta-data
- Library interface
 - Access memory through `stm_load` & `stm_store` lib. calls
 - Slow (factor of about 30)

Acceleration Alternatives

- Replace STM for small transactions
- Track some accessed locations with ASF



Use Cases – Accelerating STM

App

STM

```
stm start() {
```

SPECULATE

```
}
```

```
stm load() {
```

LOCK MOV

```
if(lock(addr))...
```

```
}
```

```
stm store() {
```

LOCK MOV

```
if(lock(addr))...
```

```
}
```

```
stm commit() {
```

COMMIT

```
}
```

Idea

- Track parts of transaction's read- and write-set using ASF

Caveats

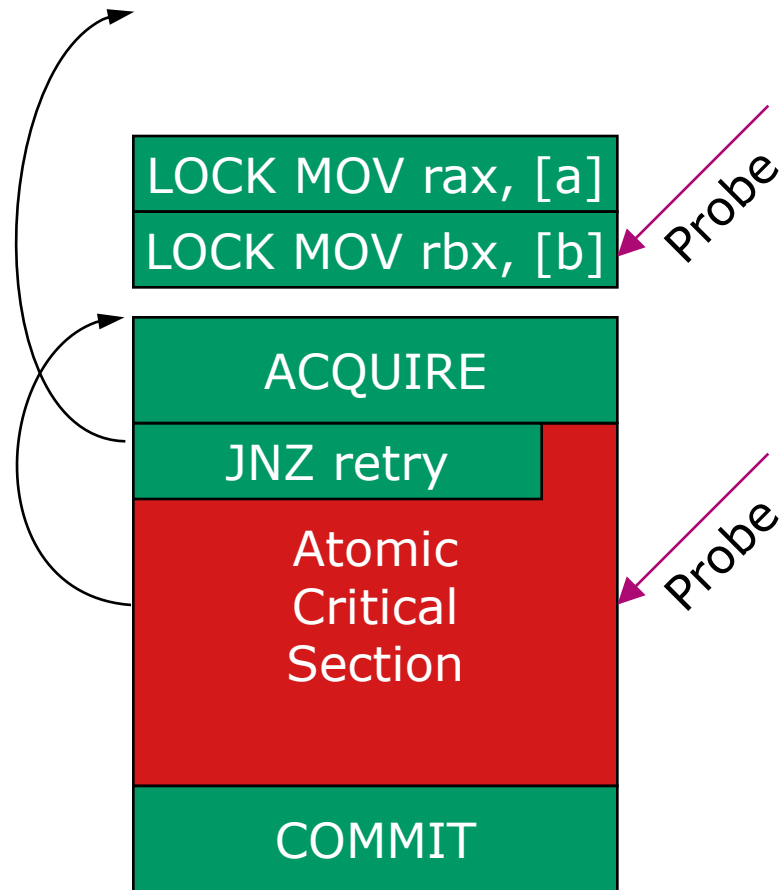
- Need to check STM's lock
- Deal with rollback
 - Destination
 - Abort STM part



Evaluation



Evaluation – ASF 1.0



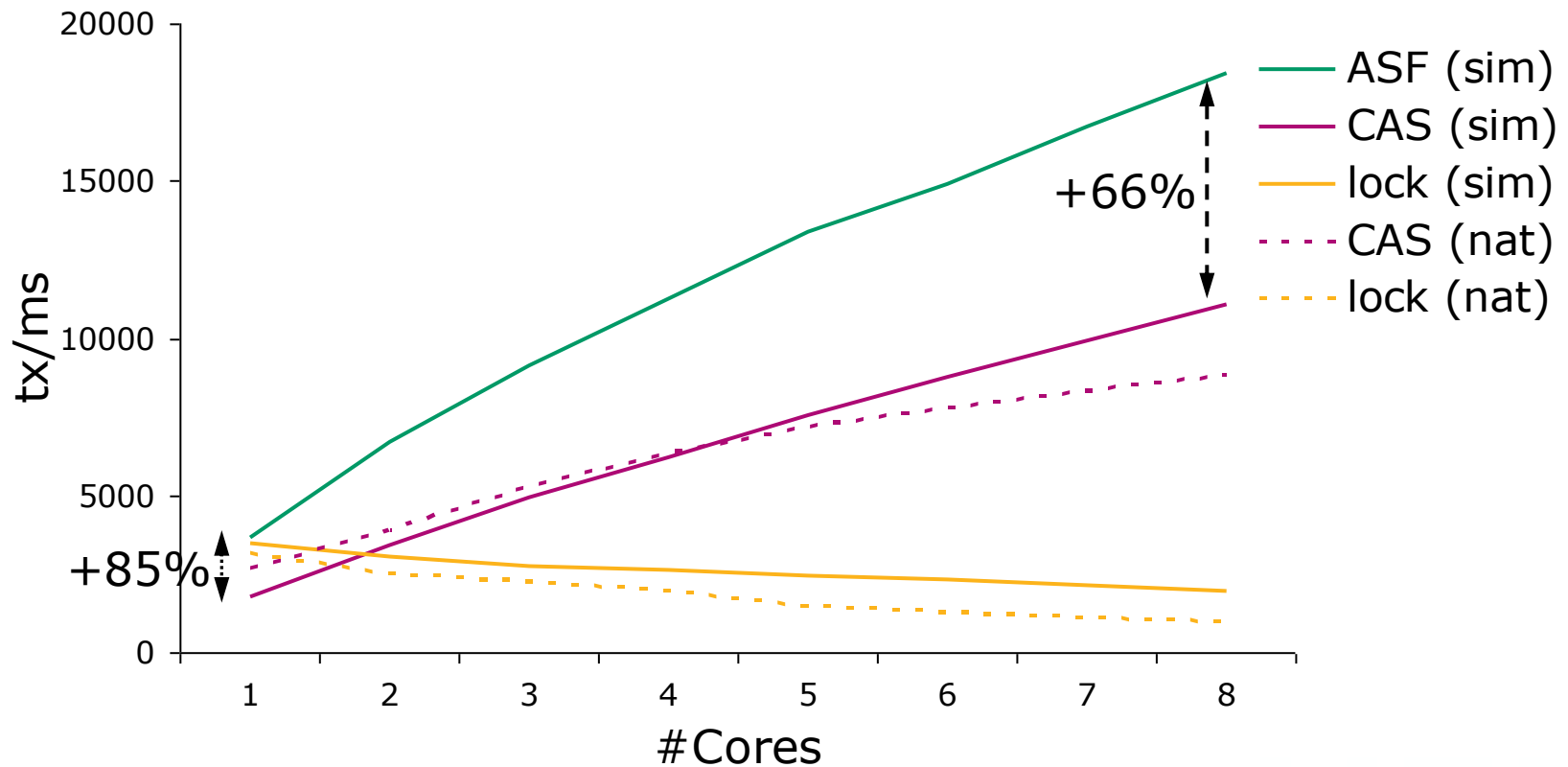
Evaluation of/with older ASF spec ("ASF 1.0")

- Differences
 - Separation: Declarators ("Spec. Phase") – Critical Section
 - No roll-back in Specification Phase
 - 8 lines capacity
- Stable at that time
- Provided helpful feedback for forging the current version



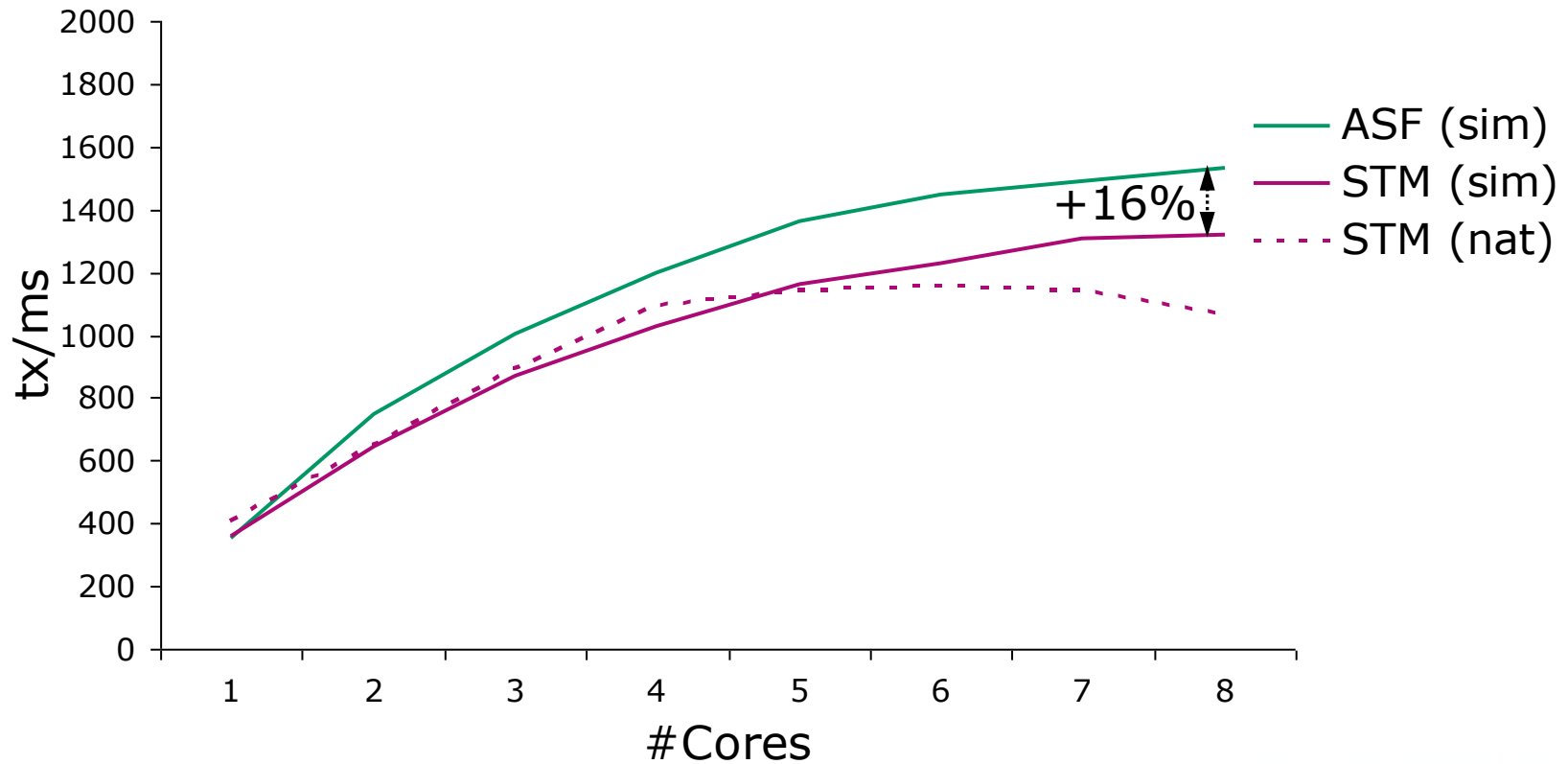
Evaluation – Bare ASF

Lock-free Throughput



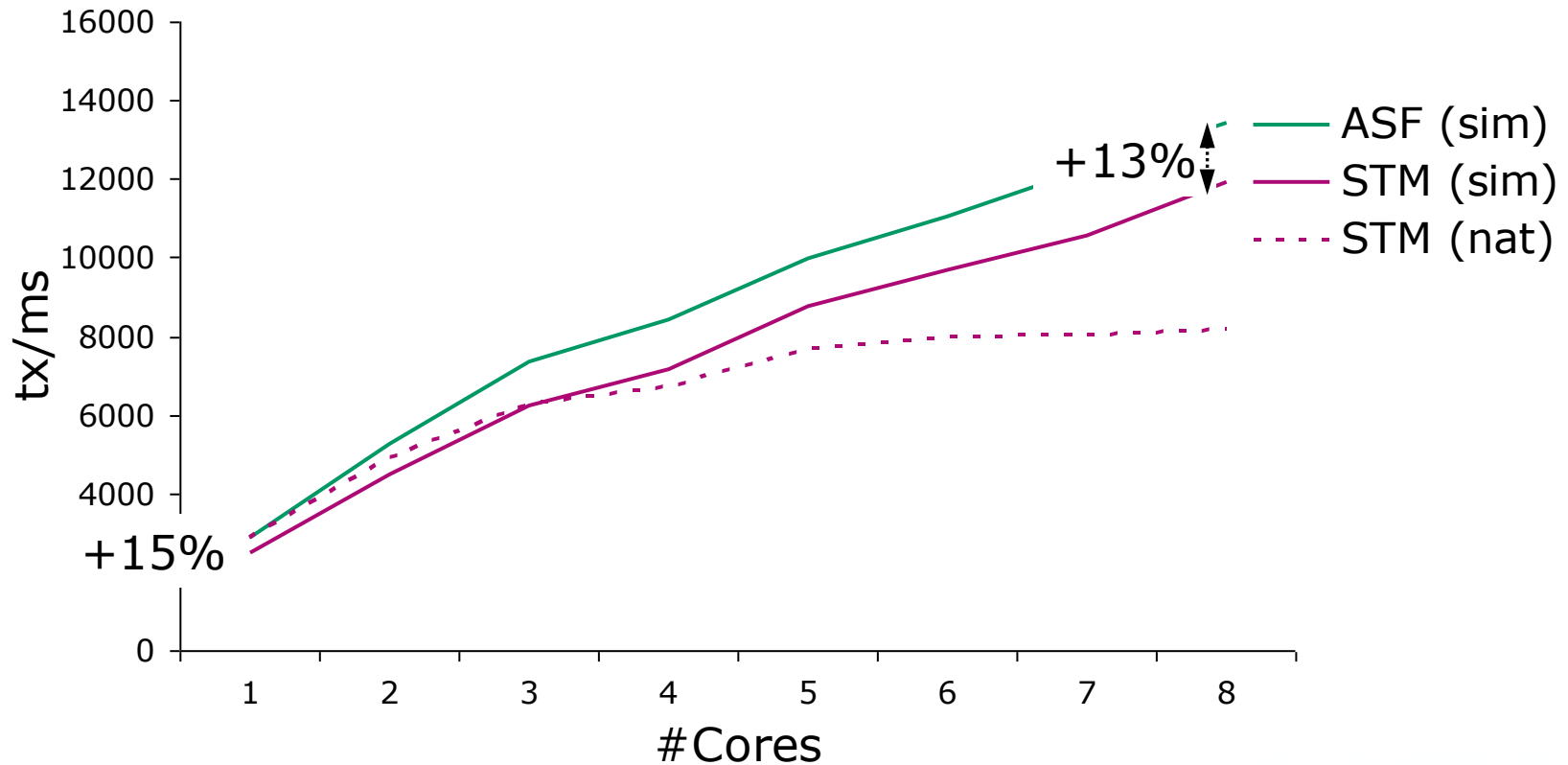
Evaluation – ASF in STM

STM - Linked List



Evaluation – ASF in STM

STM - RB-Tree (256)



Summary / Discussion



Summary

ASF is useful

- Simplifies and accelerates lock-free algorithms
- Some benefit for STM (ASF 1.0)
- Further potential with new version
- Timing simulator available

Open issues

- Explore more use cases
- Implementation in our simulators



Current Status

Full ASF 2.0 specification and simulator to be released soon

Simulator (ASF 1.X), paper and thesis available

- <http://www.amd64.org/publications.html>
- Full-system simulator PTLsim/ASF
 - Evaluate ASF inside kernel

Supported by European Union project VELOX

- <http://www.velox-project.eu>



Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2008 Advanced Micro Devices, Inc. All rights reserved.

This work was generated in the framework of the VELOX Integrated Project, which is co-financed by the European Commission through the contract no. 216852

