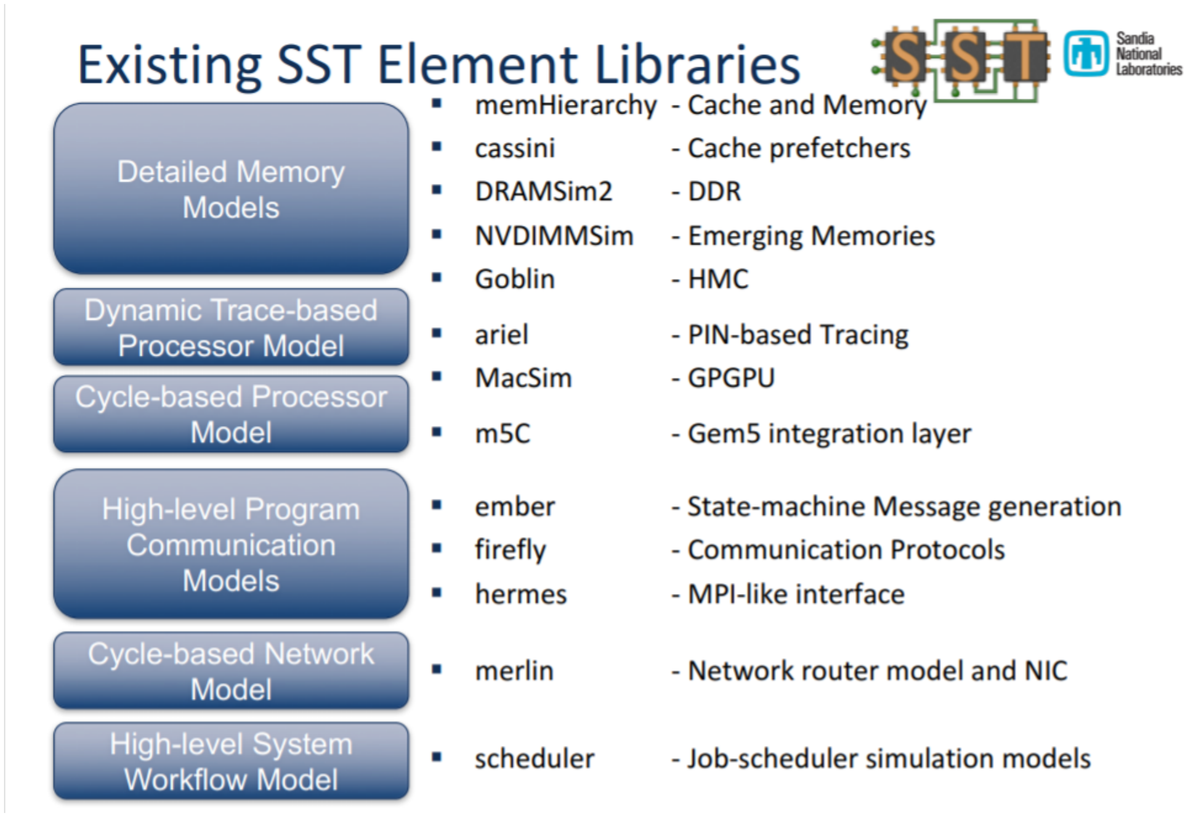
Overview

Created By	Magic Griffin
Last Edited	@Jul 05, 2019 11:24 PM
Tags	

Brief

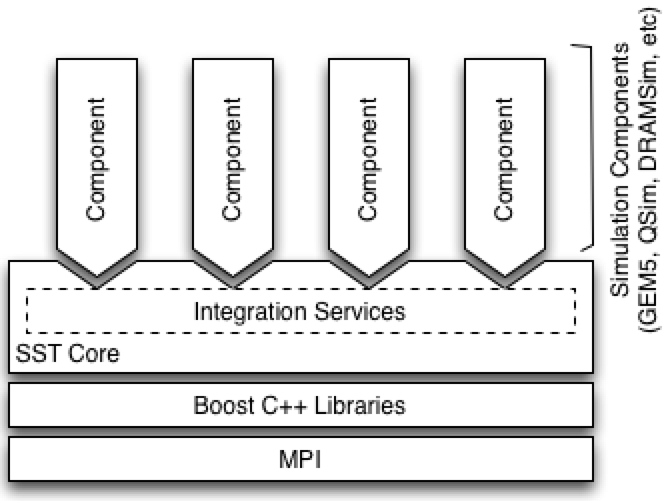
- Propose
    - Exploring novel computer system designs
- problems
    - serial
    - lack of scalability and flexibility
- Technical Approach
    - compiler tools, well defined ISA features and complete microarchitectural definitions ...
    - Parallel - MPI + Threads
    - Flexible - simulation components
    - Open API - easily extensible with new models



Existing SST Element Libraries

Detailed Memory Models	memHierarchy	- Cache and Memory
	cassini	- Cache prefetchers
	DRAMSim2	- DDR
	NVDIMMSim	- Emerging Memories
	Goblin	- HMC
Dynamic Trace-based Processor Model	ariel	- PIN-based Tracing
	MacSim	- GPGPU
Cycle-based Processor Model	m5C	- Gem5 integration layer
High-level Program Communication Models	ember	- State-machine Message generation
	firefly	- Communication Protocols
	hermes	- MPI-like interface
Cycle-based Network Model	merlin	- Network router model and NIC
High-level System Workflow Model	scheduler	- Job-scheduler simulation models

- component

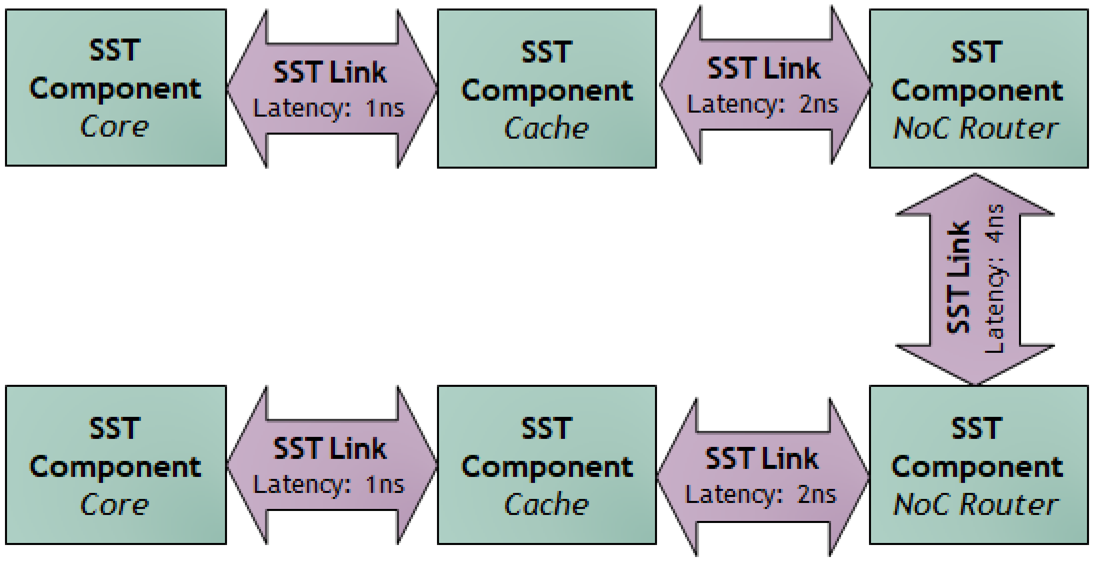


- core - contain the simulation engine and API interface to simulation elements
  - clocks, event exchange, statistics, parameter management, parallelism support ….

- elements
    - memory, cache, processor architecture, networking …
- macro - functionality for simulating extreme-scale applications

High-level View

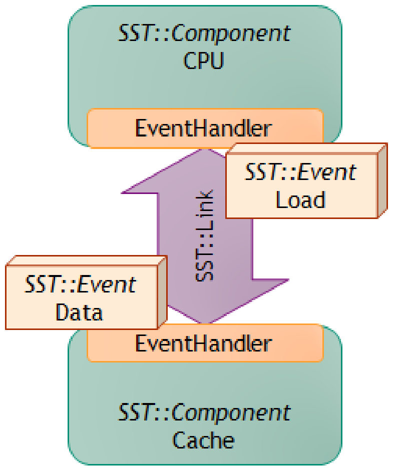
- comprised of components connected by ***links***
- Components define ***ports*** which are valid connection points for a link
- Components can use ***subComponents*** and ***modules*** for extra functionality



12

Key objects

- SST::Component
  - Simulation model
- SST::Link
  - Communication path between two components
  - Poll or interrupt with EventHandler
- SST::Event
  - A discrete event
- SST::Clock::Handler
  - Function to handle a clock tick
- SST::SubComponent
  - Add functionality to Components

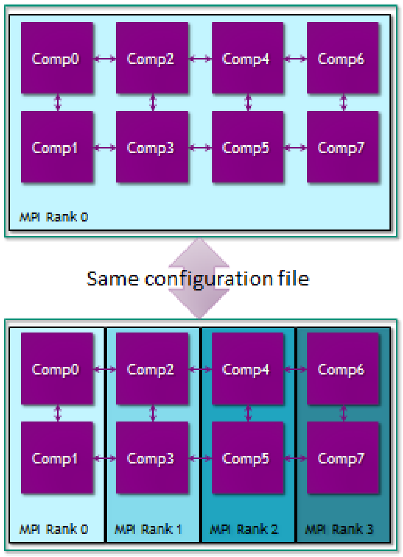


- Polled: Register a clock handler to poll the link
- Interrupt: Register an event handler to be called when an event arrives

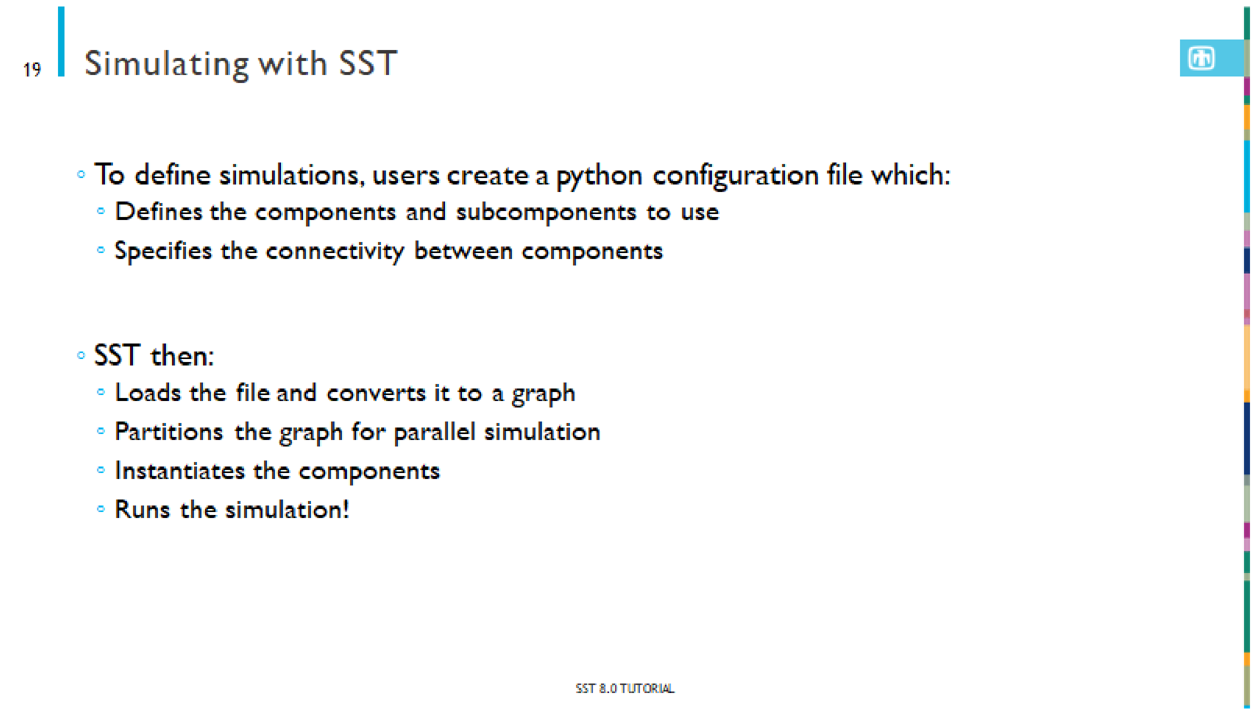
22

SST in Parallel

- SST was designed from the ground up to enable scalable, parallel simulations
- Components distributed among MPI ranks
  - And/or threads
- Links allow parallelism
  - Hence, components should communicate via links only
  - Transparently handle any MPI/thread communication
  - Specified link-latency determines synchronization rate



Case Study



19 Simulating with SST

- To define simulations, users create a python configuration file which:
  - Defines the components and subcomponents to use
  - Specifies the connectivity between components

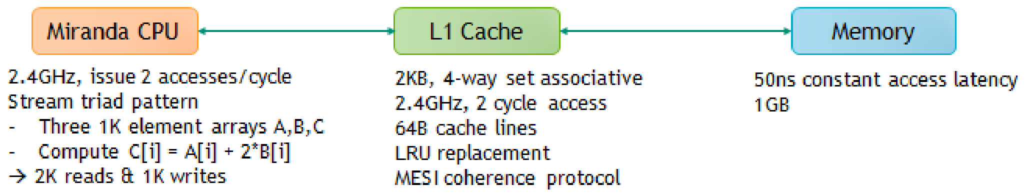
- SST then:
  - Loads the file and converts it to a graph
  - Partitions the graph for parallel simulation
  - Instantiates the components
  - Runs the simulation!

SST 8.0 TUTORIAL

26

Running a simulation: Example

- We'll walk through the configuration file, then run it
- Full intro to the elements is next but until then this example uses:
  - **Miranda** – Pattern-based memory access generator
    - With a *stream* memory access pattern generator
  - **MemHierarchy** – Caches and memory models
    - Just an L1 cache and memory for this example



2.4GHz, issue 2 accesses/cycle
Stream triad pattern
- Three 1K element arrays A,B,C
- Compute C[i] = A[i] + 2*B[i]

→ 2K reads & 1K writes

2KB, 4-way set associative
2.4GHz, 2 cycle access
64B cache lines
LRU replacement
MESI coherence protocol

50ns constant access latency
1GB

SST 8.0 TUTORIAL

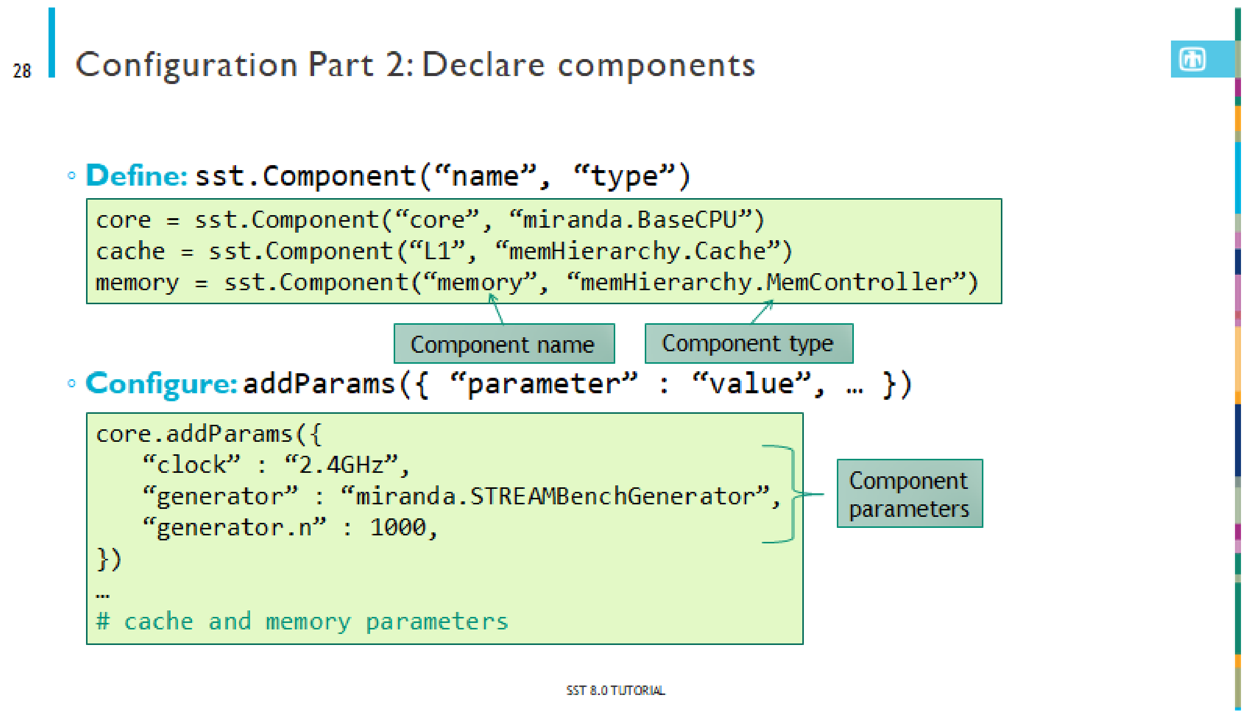
27

Configuration Part 1: Configure SST

- Set any global simulation parameters

- `sst.setProgramOption(“stopAtCycle”, “100ms”)`
  - Kill simulation (nicely!) if it runs to 100ms

SST 8.0 TUTORIAL


28

Configuration Part 2: Declare components

- **Define:** `sst.Component("name", "type")`

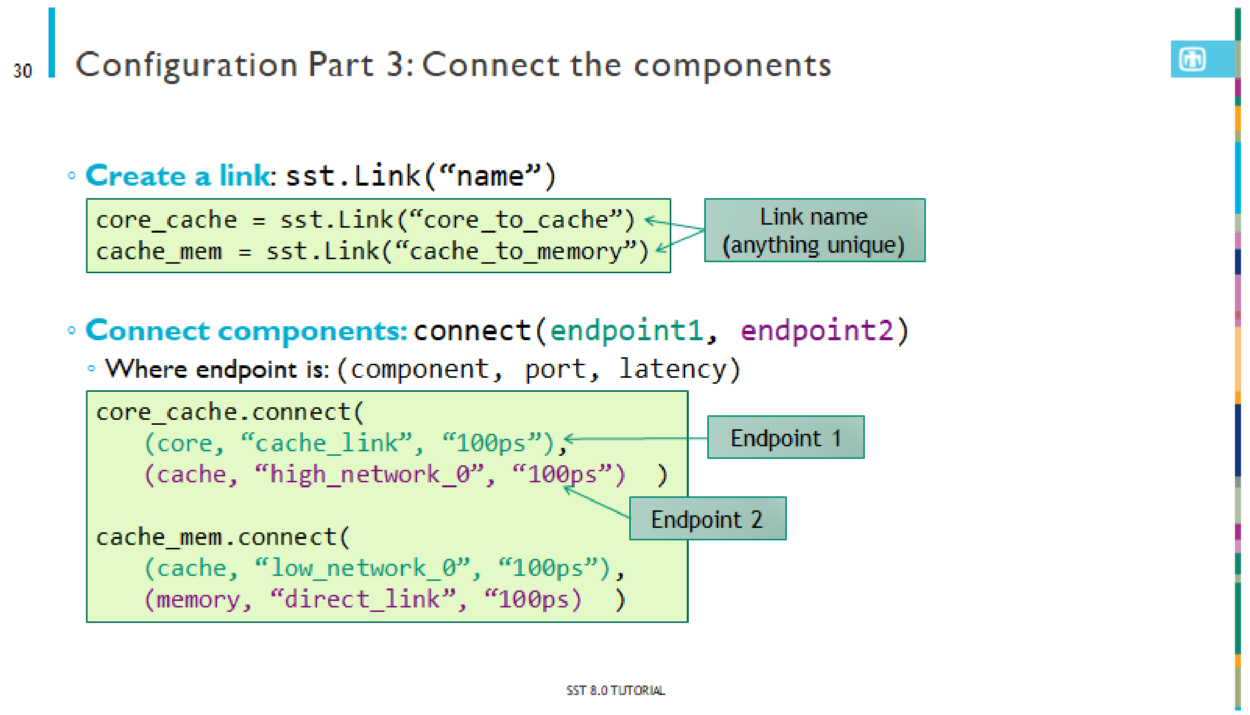
```
core = sst.Component("core", "miranda.BaseCPU")
cache = sst.Component("L1", "memHierarchy.Cache")
memory = sst.Component("memory", "memHierarchy.MemController")
```

Component name
Component type

- **Configure:** `addParams({ "parameter" : "value", … })`

```
core.addParams({
    "clock" : "2.4GHz",
    "generator" : "miranda.STREAMBenchGenerator",
    "generator.n" : 1000,
})
…
# cache and memory parameters
```

SST 8.0 TUTORIAL


30

Configuration Part 3: Connect the components

- **Create a link:** `sst.Link("name")`

```
core_cache = sst.Link("core_to_cache")
cache_mem = sst.Link("cache_to_memory")
```

Link name
(anything unique)

- **Connect components:** `connect(endpoint1, endpoint2)`
  - **Where endpoint is:** `(component, port, latency)`

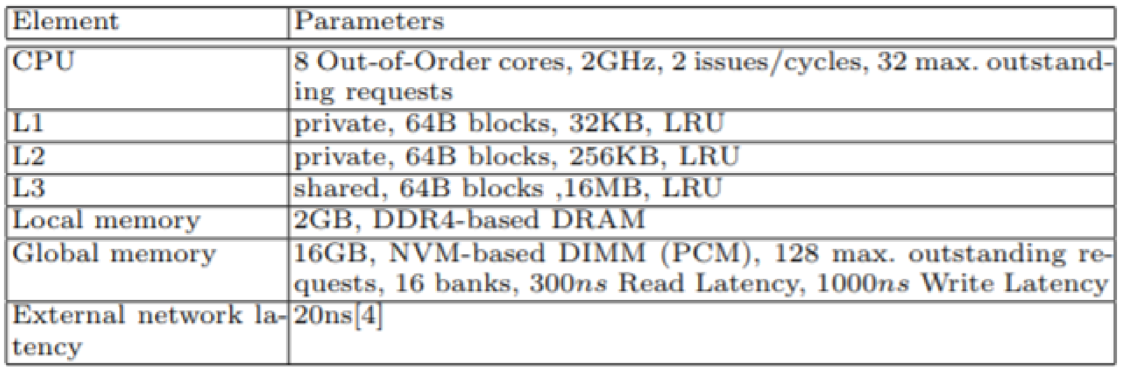
```
core_cache.connect(
    (core, "cache_link", "100ps"),
    (cache, "high_network_0", "100ps")   )

cache_mem.connect(
    (cache, "low_network_0", "100ps"),
    (memory, "direct_link", "100ps)   )
```

SST 8.0 TUTORIAL

- Language
  - Python

    - C/C++
- Required Tools(?)
    - MPI
    - Boost C++ Libraries
- Critics?
    - https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/index1.pdf

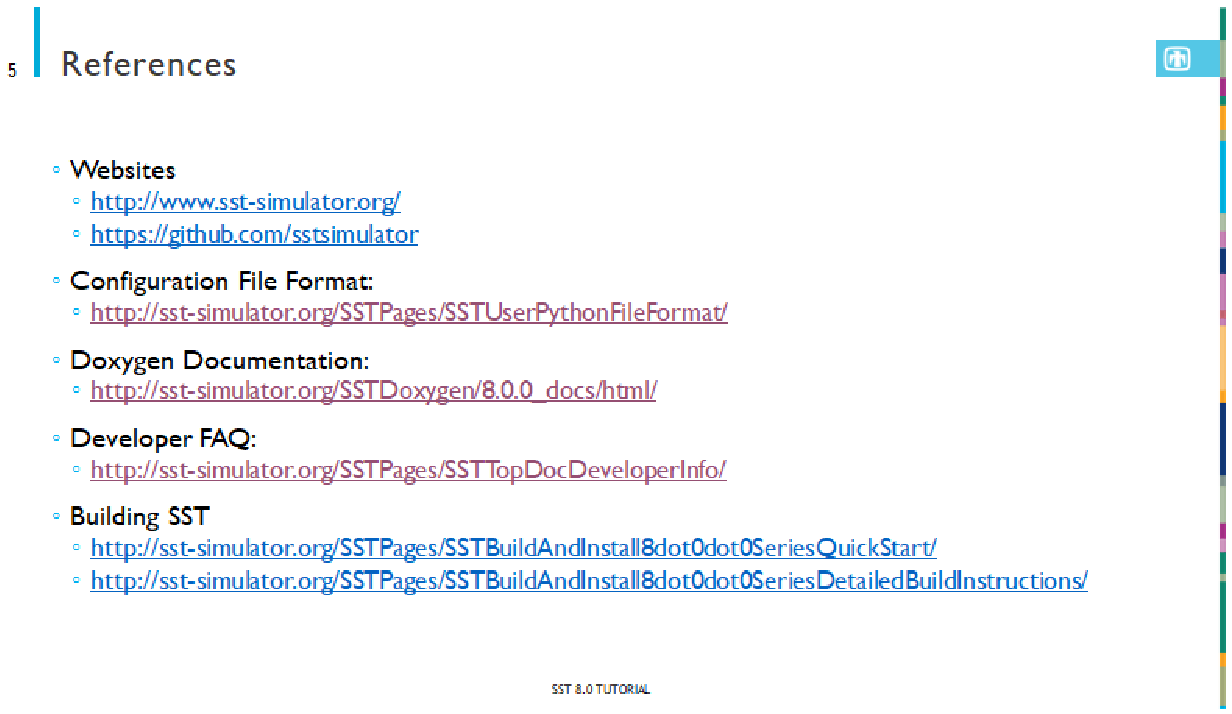
Table 1. Simulation Parameters



Element	Parameters
CPU	8 Out-of-Order cores, 2GHz, 2 issues/cycles, 32 max. outstanding requests
L1	private, 64B blocks, 32KB, LRU
L2	private, 64B blocks, 256KB, LRU
L3	shared, 64B blocks ,16MB, LRU
Local memory	2GB, DDR4-based DRAM
Global memory	16GB, NVM-based DIMM (PCM), 128 max. outstanding requests, 16 banks, 300ns Read Latency, 1000ns Write Latency
External network latency	20ns[4]

- Potential Direction
    - developing SST Element Libraries
    - ....

References


5

References

- Websites
  - http://www.sst-simulator.org/
  - https://github.com/sstsimulator
- Configuration File Format:
  - http://sst-simulator.org/SSTPages/SSTUserPythonFileFormat/
- Doxygen Documentation:
  - http://sst-simulator.org/SSTDoxygen/8.0.0_docs/html/
- Developer FAQ:
  - http://sst-simulator.org/SSTPages/SSTTopDocDeveloperInfo/
- Building SST
  - http://sst-simulator.org/SSTPages/SSTBuildAndInstall8dot0dot0SeriesQuickStart/
  - http://sst-simulator.org/SSTPages/SSTBuildAndInstall8dot0dot0SeriesDetailedBuildInstructions/

SST 8.0 TUTORIAL

https://hpc.pnl.gov//modsim/2017/Presentations/SST_Modsim17.pdf

SST Publications
http://sst-simulator.org/SSTPages/SSTTopDocPublications/