Student Cluster Competition 2019
The Structural Simulation Toolkit (SST)

Workload 1 (60/100 SST points)
Scenario
The research department at your company has developed a new application, but it runs terribly on their outdated hardware, so they've been given $5000 to purchase a new machine. As the chief computer architect, they've asked you to advise them on what to buy. You have received some quotes for different systems, but there are many options to pick from and each has a different cost (see table below). Further, there's some disagreement on which system would be best: the research department wants the fastest system for the application, while the finance department wants the most cost efficient system (highest performance per cost). You need to use SST to determine which two configurations, within the budget, meet the goal of each department (highest performance and highest performance/cost). Hopefully they will agree on the definition of best by then, and if not, you can sit back and relax while they argue since you'll have the answer no matter who wins. To help you, the research department has modeled their workload as a Miranda workload generator subcomponent.

Instructions

1. Install **SST 9.1** (not 9.0!) from www.sst-simulator.org. You will need sst-core and sst-elements.
2. Edit the Makefile in the wk1-miranda/ directory so that SSTELEMSOURCE and SSTCOREINSTALL point to your sst-elements source root directory and sst-core install root directory, respectively. Then run `make.` You can run '`sst-info sc19`' to check that SST can now find the new subcomponent.
3. The SST input script *wk1-miranda/scc-sst-node.py* is parameterized to take each of the system options in the table. A comment at the top of the script describes the simulated architecture in some more detail. If you modify the script, do not modify the simulated architecture or workload (i.e., specifying SST partitioning or printing additional information is OK). The script can be run as follows:

```
sst scc-sst-node.py --model-options=”-n=X0 -c=X1 -t=X2 -l1=X3 -l2=X4 -s=X5 -l3=X6 -b=X7 -w=X8 -m=X9”
```

   where the **X**s should be replaced with an option from the table. For example:

```
sst scc-sst-node.py --model-options=”-n=40 -c=fast -t=no -l1=big -l2=small -s=private -l3=small -w=6
-b=slow -m=basic”
```

4. The script will immediately reject and not run any configuration that exceeds the $5000 limit with the message "ABORT: Cost exceeds limit of 5000. Cost is …". If the --model-options string you give is valid and within the budget, the script will print the configuration and cost, and the simulation will run.
5. Two text files have been provided in this directory for you to record your answers. Fill in the configuration (--model-options string), simulated time, and cost for the highest performing system (wk1-perf.txt) and highest performance/cost system (wk2-cost.txt). Correct answers receive all 30 points per answer; incorrect answers receive up to 25 points prorated by how closely they come to the correct system. Invalid configurations, configurations that exceed the budget, and answers with an incorrect cost or simulation time will receive no points. For simulation time, include all digits and units reported by SST (e.g., "32.3187 us").

***Blue** indicates arguments the script accepts. For example, the options for core type are -c=slow, -c=medium, or -c= fast.*

Category	**Script param**					
Core count	-n	22	24	30	36	40
Core type	-c	slow	medium	fast		
		$15 / core	$23 / core	$40 / core		
		1.8GHz 16 outstanding memory requests	2.5GHz 16 outstanding memory requests	4GHz 32 outstanding memory requests		
SMT	-t	no	yes			
		free	1.7 * core cost			
		1 hardware thread/core	2 hardware threads/core			
L1 size (per core)	-l1	small	big			
		$18 / core	$28 / core			
		16KB, 8-way, 1 cycle access	64KB, 16-way, 4 cycle access			
L2 size (per core)	-l2	small	big			
		$14 / core	$20 / core			
		128KB, 8-way, 5 cycle access	512KB, 16-way, 7 cycle access			
L2 organization	-s	private	shared			
		free	$2 / core			
		private	shared			
L3 size (per core)	-l3	small	big			
		$20 / core	$24 / core			
		1MB, 16-way, 10 cycle access	1.5MB, 32-way, 14 cycle access			
Network-on-chip	-b	slow	fast			
		$5 / core	$9 / core			
		1.6GHz	2.2GHz			
Memory channels	-w	6	8			
Memory type	-m	basic	bw	lat		
		$110 / channel	$200 / channel	$260 / channel		
		Baseline	2X bandwidth	~1/3 lower latency		

Workload 2 (30/100 SST points)

Scenario

Although the finance and research departments still haven't agreed on the definition of best for the new $5000 machine, one of the research department's sister groups has inexplicably been given the go ahead to procure an 8000+ node cluster. The only technical detail they still need to nail down is the decision between a dragonfly and hyperx network topology. Since they know the research department has applications that could benefit from running on a large system, they are soliciting your input on which network to procure. The research department has already created an Ember description of their workload and the sister group has provided platform files describing the two possible network configurations. All you need to do is run the models on SST and report performance data. The data is needed as quickly as possible in order to be considered in the final deliberations.

Instructions

1. If not done already, install **SST 9.1** (not 9.0!) core and elements from www.sst-simulator.org.
2. The workload is described in the *wk2-ember/scc-sst-interconnect-workload.load* file. This file will be used in conjunction with two platform files (in wk2-ember/): *dragonflyParams.py* and *hyperxParams.py,* which are used to describe the possible network configurations.
3. The command-lines needed to run these configurations are found in the *wk2-ember/scc-sst-interconnect-dragonfly.sh* and *wk2-ember/scc-sst-interconnect-hyperx.sh* scripts. You will need to ensure that the sst/elements/ember/test directory is included in your PYTHON_PATH. You may change the parallelization used (both mpi ranks and threads) to provide what you feel will be the best performance.
4. The output from the runs should be piped to files in this (scc-sst/) directory: **scc-sst-interconnect-dragonfly.out** and **scc-sst-interconnect-hyperx.out**. The workload is broken up into 10 iterations. The configuration is set to print out each time an ember motif is started or stopped, and each iteration ends with an allreduce, which is reported by a line like the following:

   Allreduce: ranks 65536, loop 1, 1 double(s), latency XXX.XXX us

   You will receive 1.5 points for each iteration correctly completed on each of the network configurations.

The interview will be worth the remaining 10/100 SST points.