PEA – FINAL PROJECT

JASKARAN RAM - 10735884 - 2023

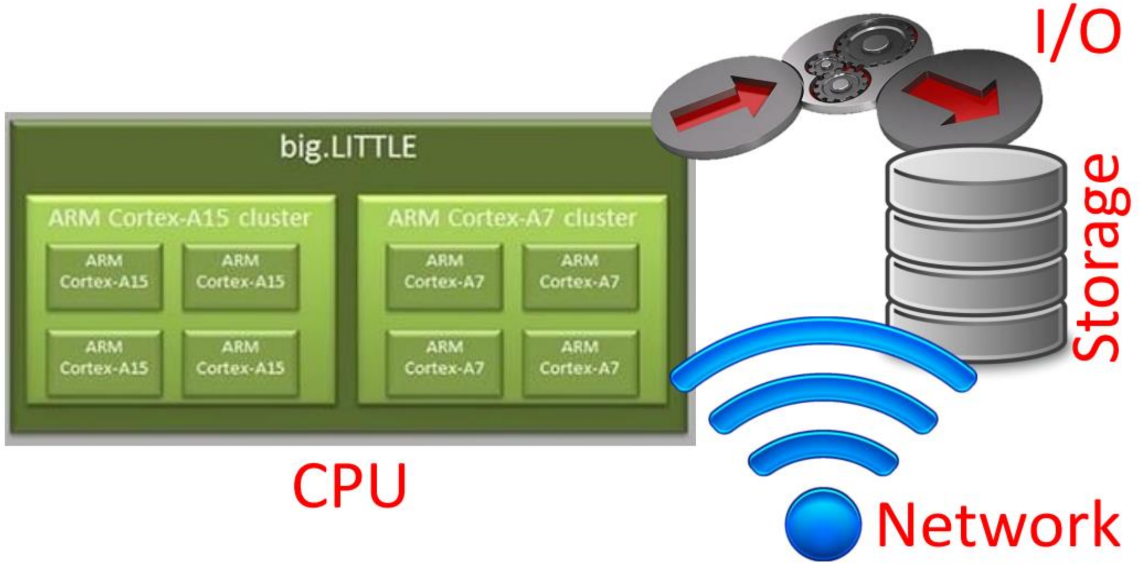
TYPE B

MARCO GRIBAUDO

GOAL OF THE PROJECT

Performance of a Big-Little architecture

- Determine the best assignment probability distribution of tasks:
  - ➢ Test a few alternatives of probabilities of assigning a heavy computation task to an efficiency core and of assigning low computation task to a high-performance core.
- Determine the system throughput in each scenario.



CHARACTERISTICS OF THE MODEL

- A system characterized by a Big-Little architecture is characterized by **4 high performance cores**, and **8 energy efficient cores.**
- It is used by **NB = 10** heavy computation tasks, and **NL = 32** low computation tasks.
- The scheduler will mainly schedule heavy computation tasks on the high-performance cores, while low computation tasks on the energy efficient cores. However, to better use the resources, there is also a small probability that tasks will be assigned the other way round
- The execution times of the tasks on the cores are collected in the following traces (all expressed in sec):

	High Performance Cores	Energy Efficient Cores
Heavy computation tasks	`TraceB-HH.txt`	`TraceB-HE.txt`
Low computation tasks	`TraceB-LH.txt`	`TraceB-LE.txt`

All these components can be considered working in processor sharing, with an exponential service time (different per type of job), whose average is described in these tables:

	I/O	Storage	Network
Heavy computation tasks	*50 msec*	*200 msec*	*5 msec*
Low computation tasks	*150 msec*	*10 msec*	*120 msec*

OTHER COMPONENTS

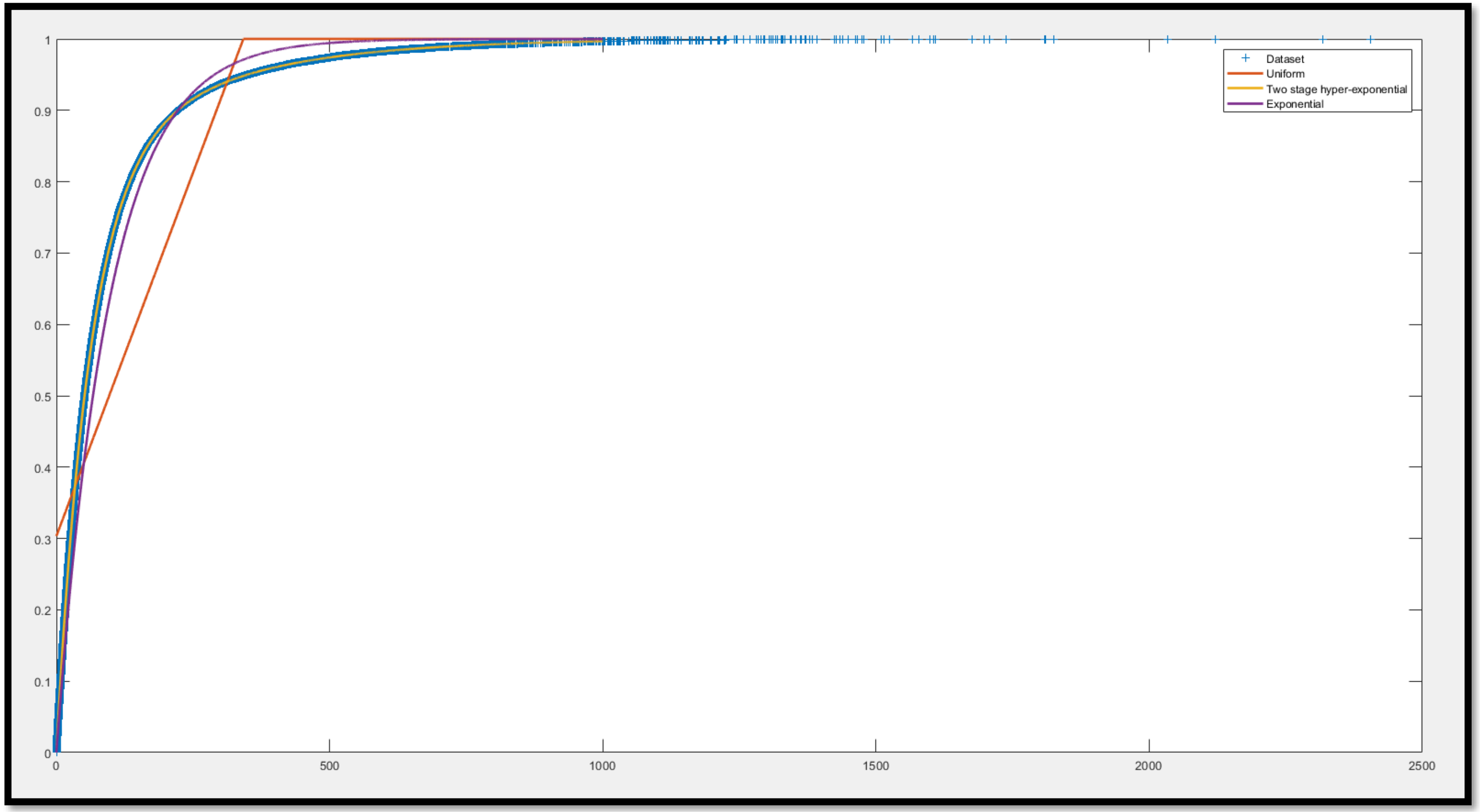
FITTING

By means of MATLAB and having at disposal samples of the durations of the corresponding execution times, I was able to define the characteristics of the various departments, exploiting the method of moments and fitting the traces according to several distributions, such as Uniform, Exponential, Hyper-Exponential, Hypo-Exponential and Erlang (they were enough to fit well). Then, looking at the coefficient of variation and at the curves in the graph I chose the most appropriate rate for each of the traces

Method of moments is used to determine the best parameters producing samples with characteristics similar to the one measured in the real trace

$$
\begin{aligned}
\mu_1 &\equiv \mathrm{E}[W] = g_1(\theta_1, \theta_2, \ldots, \theta_k), \\
\mu_2 &\equiv \mathrm{E}[W^2] = g_2(\theta_1, \theta_2, \ldots, \theta_k), \\
&\vdots \\
\mu_k &\equiv \mathrm{E}[W^k] = g_k(\theta_1, \theta_2, \ldots, \theta_k).
\end{aligned}
$$

FITTING I - HE



Heavy Computations

Energy Efficient cores

Moments:

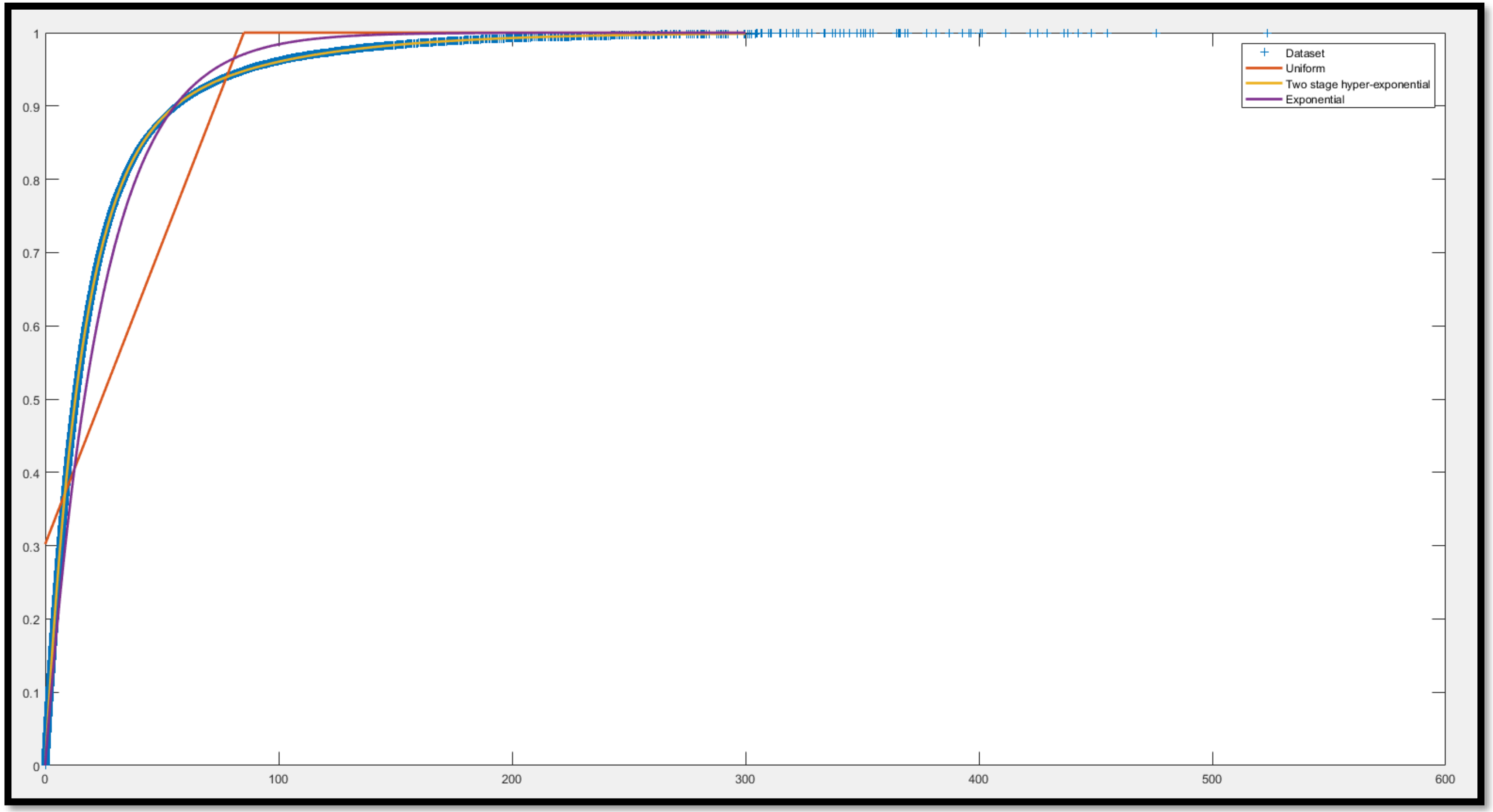
I. Moment: 96.5605
II. Moment: 29425.2
III. Moment: 1.83159e+07

CV: 1.46829

Distribution:

2 stages hyper-exponential
- p = 0.20066
- λ1 = 0.00411398
- λ2 = 0.0167277

FITTING 11 - HH



Heavy Computations

High Performance cores

<u>Moments:</u>

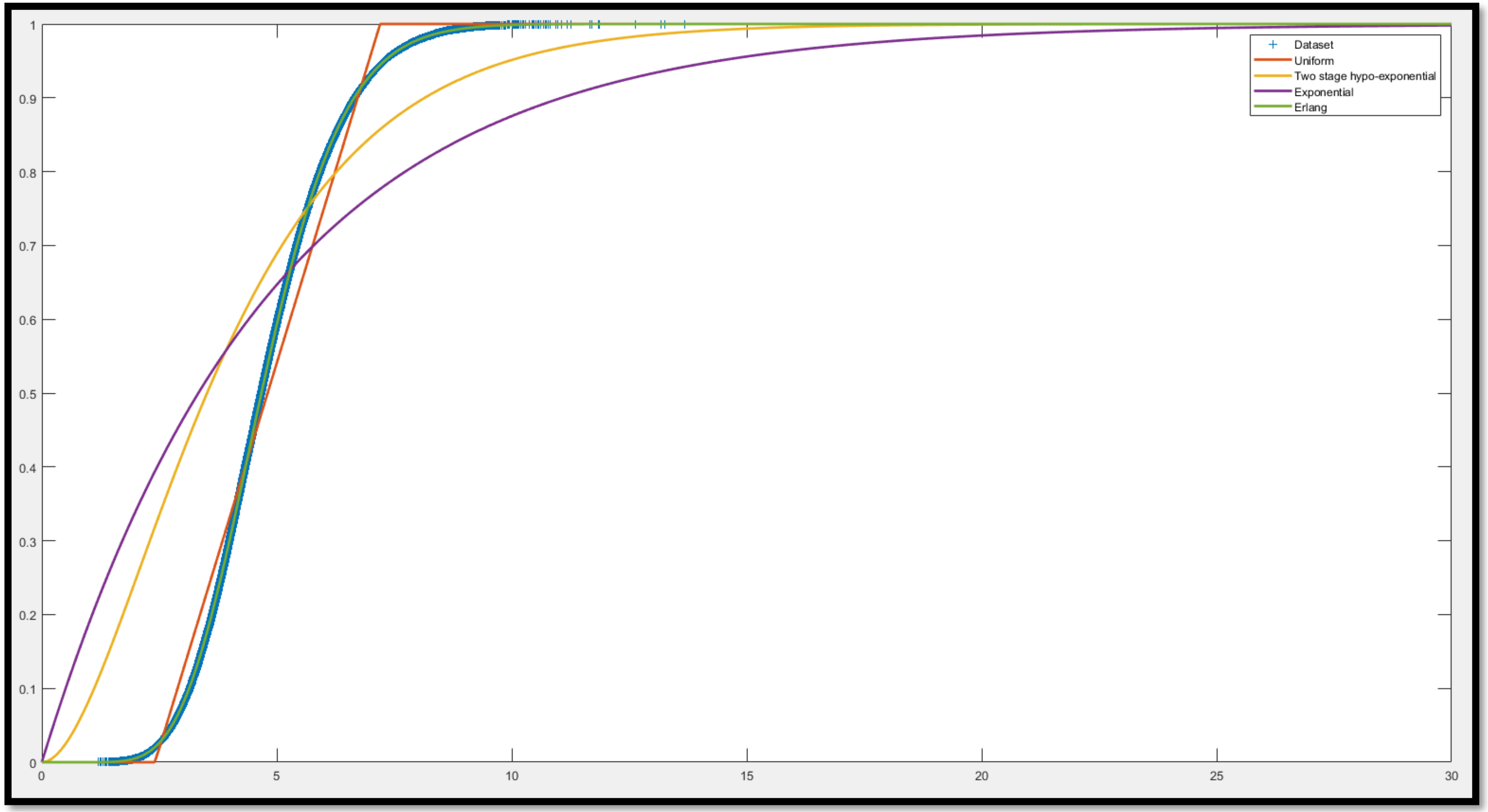
I. Moment: 24.1255
II. Moment: 1820.78
III. Moment: 278560

CV: 1.45886

<u>Distribution:</u>

2 stages hyper-exponential
- p = 0.206686
- $\lambda 1$ = 0.0167711
- $\lambda 2$ = 0.0672208

FITTING 111 - LE



Low Computations

Energy Efficient cores

Moments:

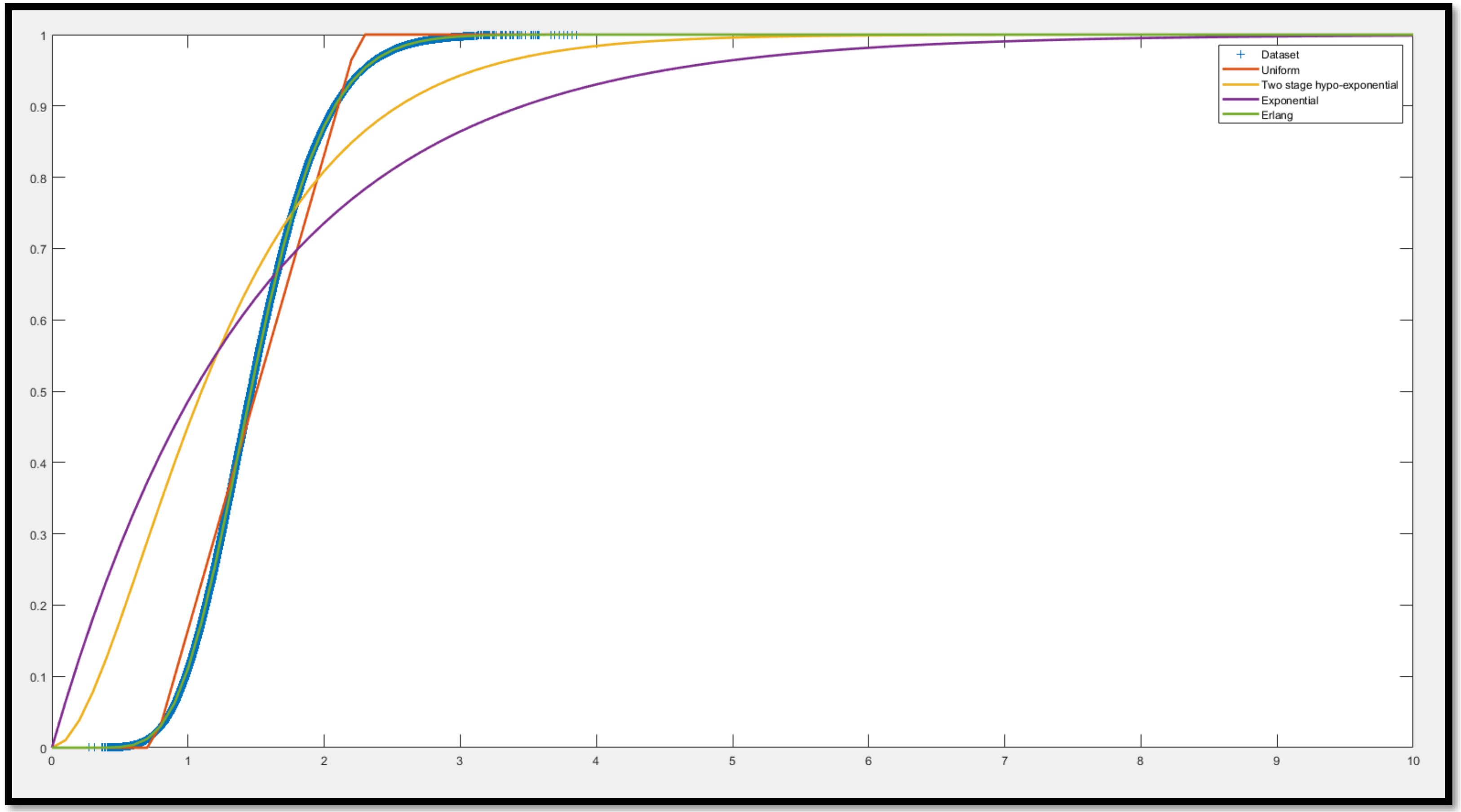
I. Moment: 4.80413
II. Moment: 24.9919
III. Moment: 139.942

➢ CV: 0.28784

Distribution:

❑ Erlang k = 12
❑ λ = 2.49785

FITTING IV - LH



Low Computations

High Performance cores

Moments:

I. Moment: 1.5025
II. Moment: 2.44536
III. Moment: 4.28698

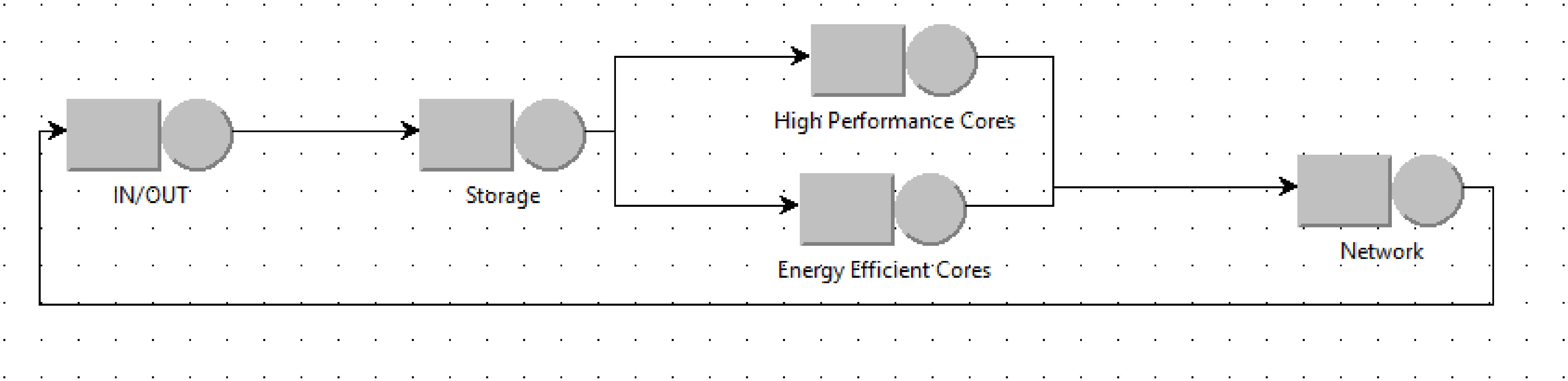
- CV: 0.288477

Distribution:

- Erlang k = 12
- λ = 7.98671

MODEL DEFINITION

- I modeled my components using JMT's JSIMgraph tool, allowing me to synthesize separable models—a system in which the order of components is irrelevant.
- All components operate under Processor Sharing with a defined Service Time. Concerning the Input/Output, Storage, and Network components, I simply expressed the parameters of the exponential distribution provided in the text. For the High Performance/Energy Efficient Cores components, I estimated the probabilistic distribution using the fitting technique, as explained in previous steps.
- As mentioned earlier, this is a closed model with precisely 10 Heavy tasks and 32 Low computation tasks, for which I created two Closed classes with their respective populations.
- The most critical aspect pertains to routing, where I evaluated various scenarios to optimize the system's Throughput (and also Response Time). In the next slides there are various scenarios studied.



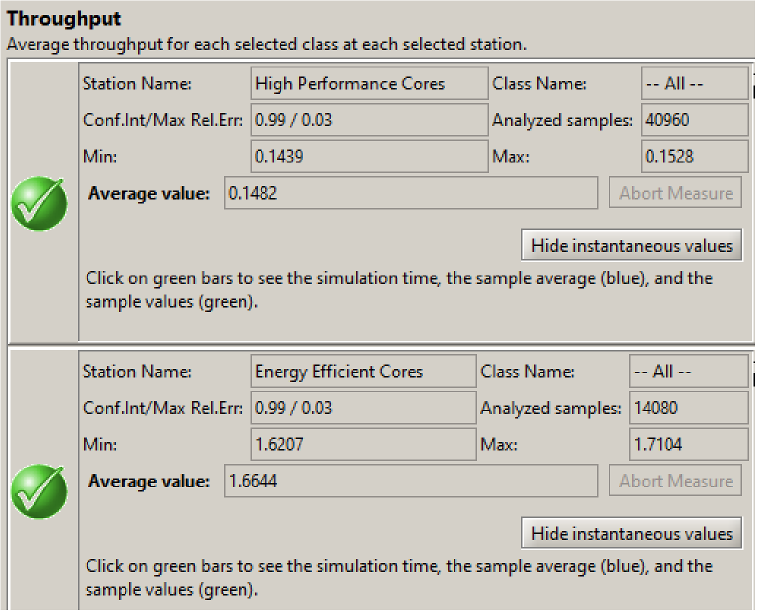
I could have added a router but it would have been superfluous since I can do the routing with any component.

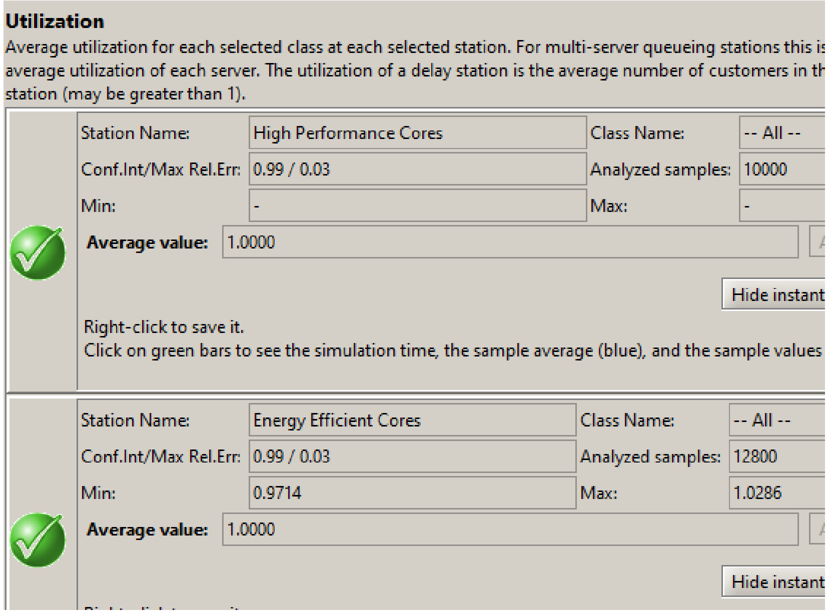
MODEL DESIGN

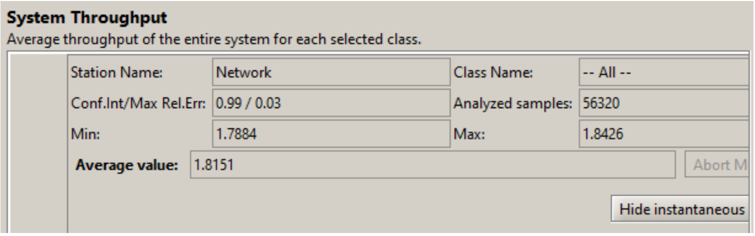
Using JMT's JSIMgraph

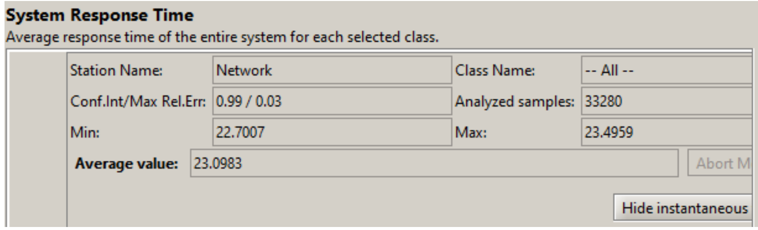
PERFORMANCE INDICES ANALYZED

<u>PROBABILITIES</u> : all Heavy Tasks on High Performance and Light Tasks on Energy Efficient





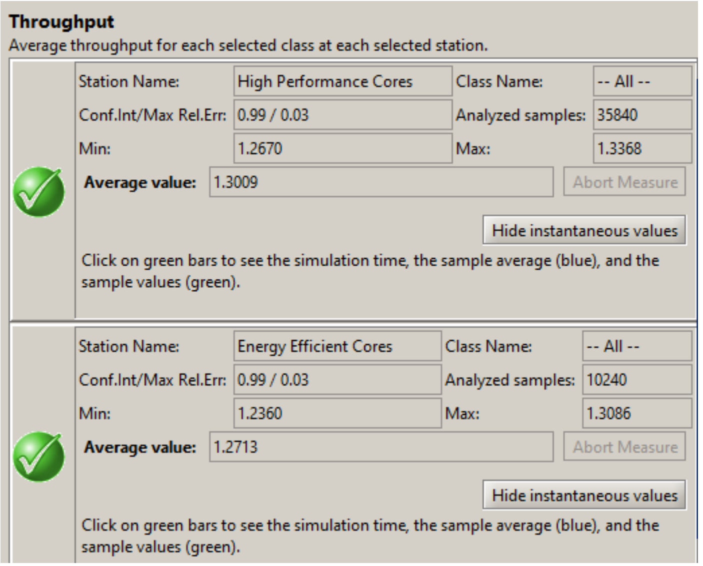


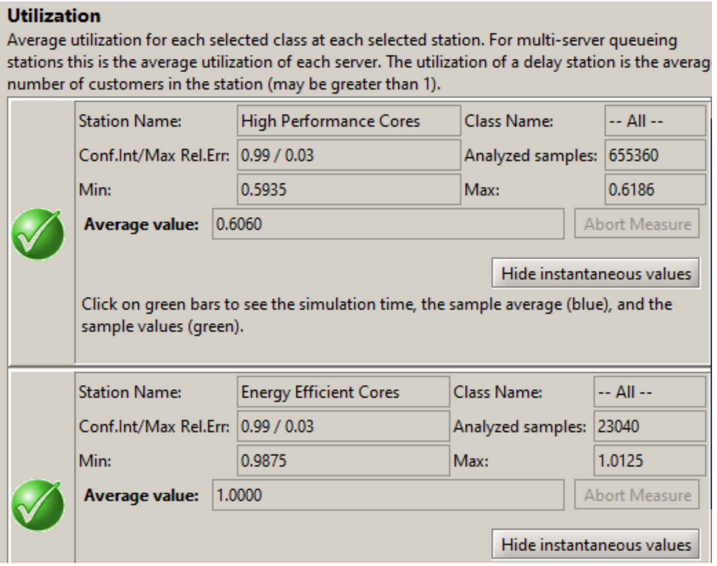


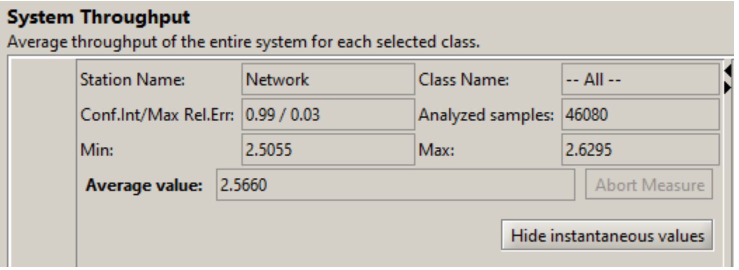
WORST ROUTING CONFIGURATION,
even it represents the main indications about routing

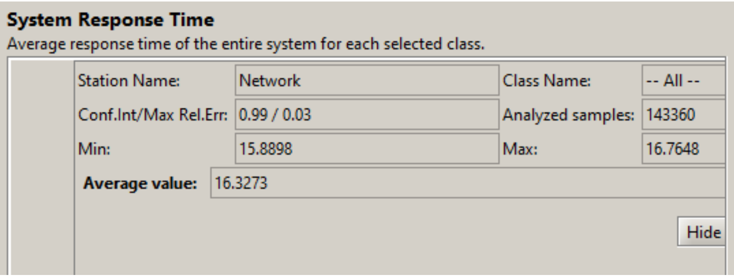
PERFORMANCE INDICES ANALYZED

RANDOM ROUTING





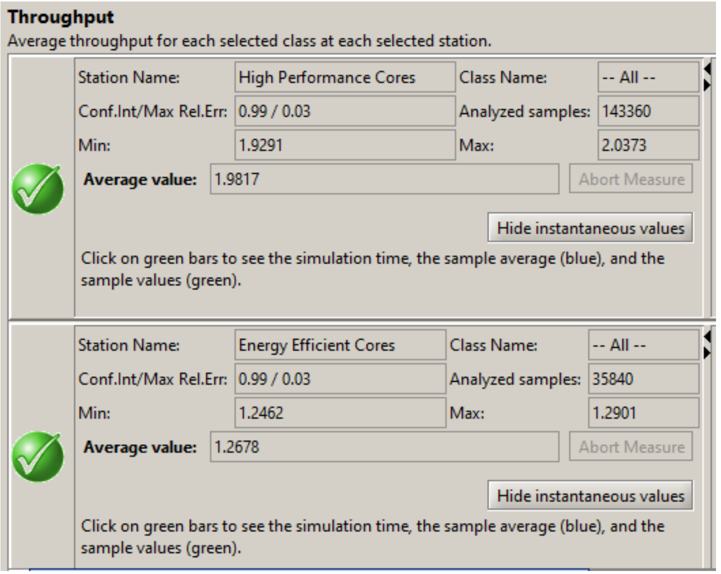


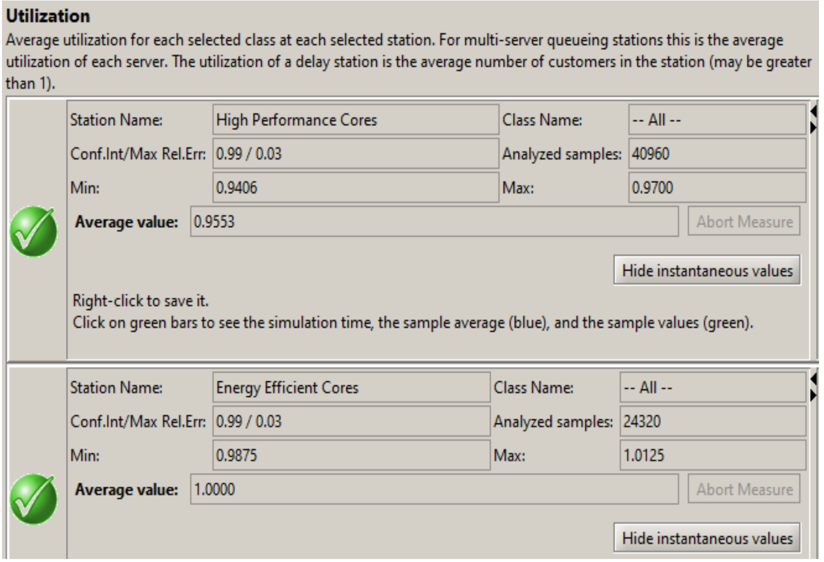


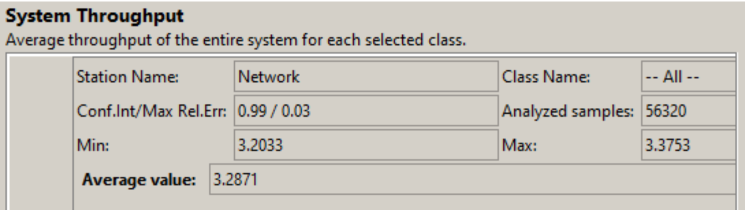
Much better performance but still not optimized (for example Utilization)

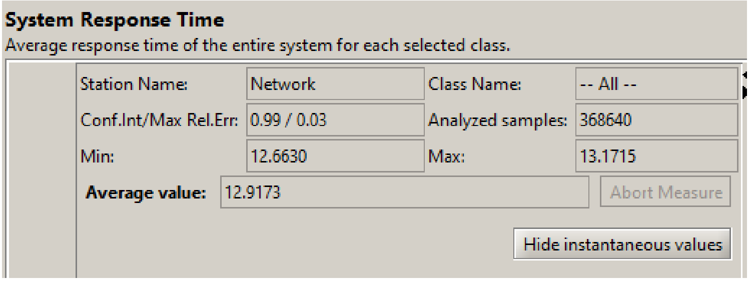
PERFORMANCE INDICES ANALYZED

LEAST UTILIZATION





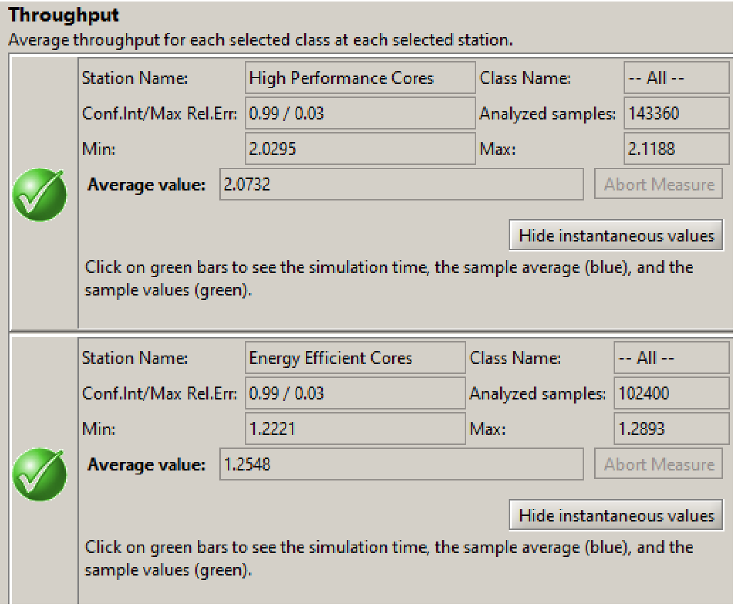


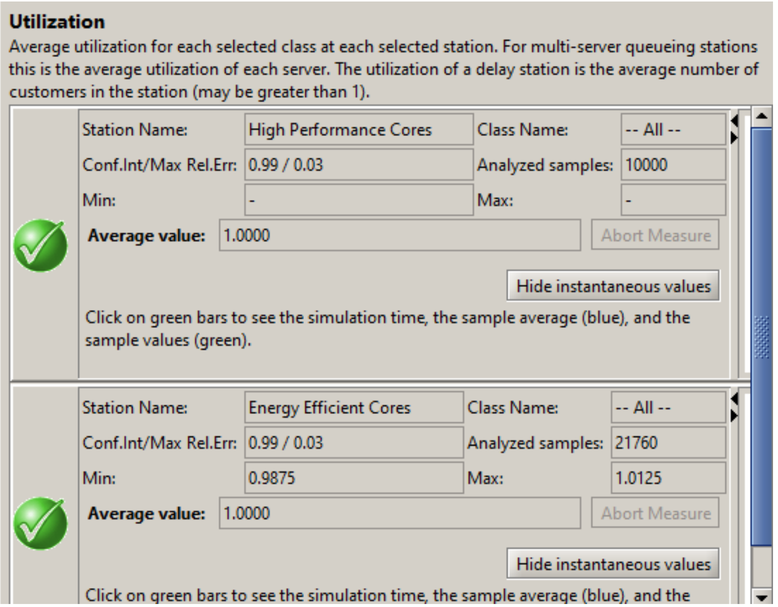


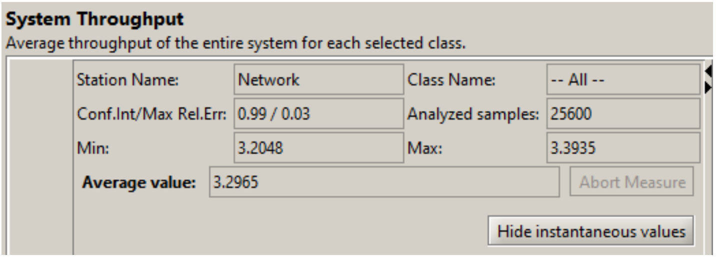
It does not comply with the routing directives but is definitely more performing

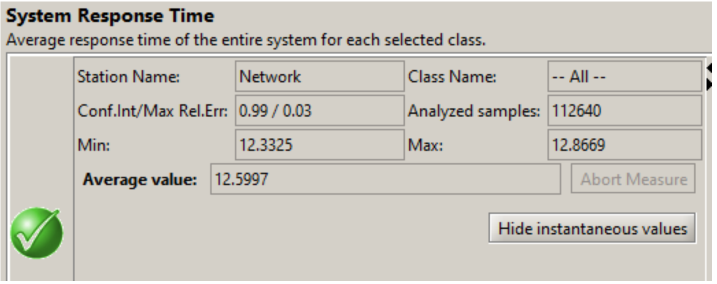
PERFORMANCE INDICES ANALYZED

<u>JOIN THE SHORTEST QUEUE</u>









BEST ROUTING CONFIGURATION!

Same annotations as last one but a slight improvement of performance

CONCLUSIONS

I have studied and analyzed four scenarios, all with a confidence interval of 99% and a maximum relative error of 3%. I have explored four different routing methods:

1. **Core Type-Based Routing:**
   - All heavy tasks are executed on high-performance cores, while light tasks are handled by energy-efficient cores (Configuration Requested but not well performing)
   - Despite achieving similar UTILIZATION, this approach doesn’t optimize Throughput.
2. **Random Routing:**
   - This method performs significantly better than the first approach but still has space for optimization.
3. **"Least Utilization" Approach:**
   - Jobs are routed to the station with the smallest utilization.
   - This configuration has proven to be one of the most efficient.
4. **"Join the Shortest Queue" Approach:**
   - This configuration has demonstrated to ensure the highest system Throughput and a better average response time.

In summary, after analyzing the four scenarios, the "Join the Shortest Queue" approach has emerged as the most promising, ensuring maximum system Throughput and optimal average response time, even it doesn’t comply with the routing directives

THANKS

JASKARAN.RAM@MAIL.POLIMI.IT