Predicting the System Performance by Combining Calibrated Performance Models of its Components

A Preliminary Study

Thomas Begin[*†] Alexandre Brandwajn[#Δ]

[*] LIP - INRIA RESO - Univ. Lyon, France
[†] DIVA Lab, Univ. Ottawa, Canada

[#] Computer Engineering - Univ. California Santa Cruz, US
[Δ] PALLAS International Corporation, San Jose, US

Outline

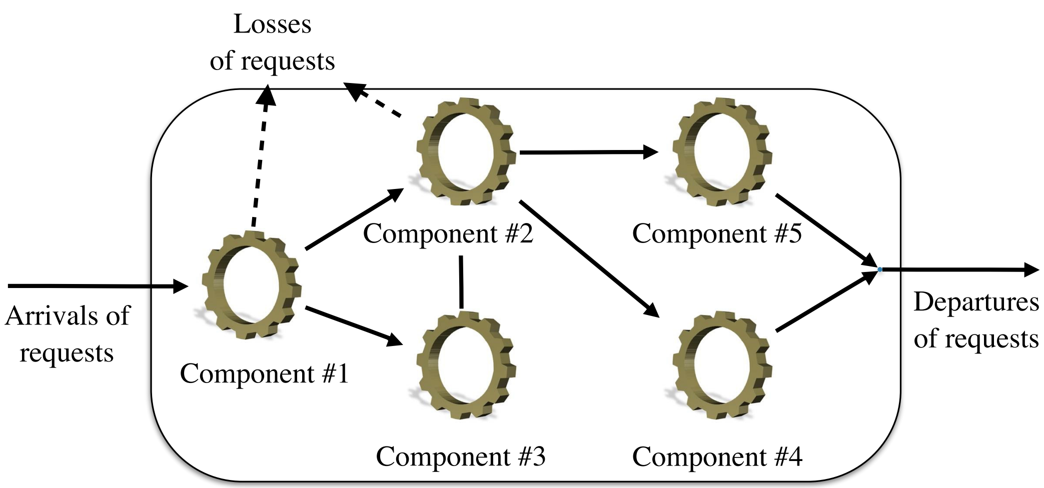
- Context and Motivation
- Proposed Approach
- Applicability Conditions
- Examples
  - Success
  - Failure
  - Unknown component
- Conclusions

Constructive Modeling

- Computer and telecommunication systems
  - Increasingly complex architectures
  - Several components
  - Unknown aspects
- Performance modeling more difficult, but no less important
- Classical approach: constructive modeling
  - "Mimic" the structure of the system
  - Expertise for building and solving the model
- What if this approach doesn't seem applicable?

Today's issue

- Using only the performance of each individual component
- How to obtain the performance of the whole system?



Open system with 5 components

Proposed Approach

Assuming calibrated models for each of the *K* components of an open system

- The mean number of requests in the whole system
  - straightforward
- The mean response time for the whole system
  - from Little's law $$r_S^{mod} = \frac{\sum_{k=1}^{K}(r_k^{mod} x_k^{mod})}{x_S^{mod}}$$
- The loss probability for the whole system
  - more involved $$p_S^{mod} = \frac{1}{1 + \frac{x_S^{mod}}{\sum_{k=1}^{K} x_k^{mod} . p_k^{mod} / (1 - p_k^{mod})}}$$

Applicability Conditions

- Assumptions

  1) The throughput ratios constant or known

     - The relationship between the overall system throughput and the throughputs of individual components

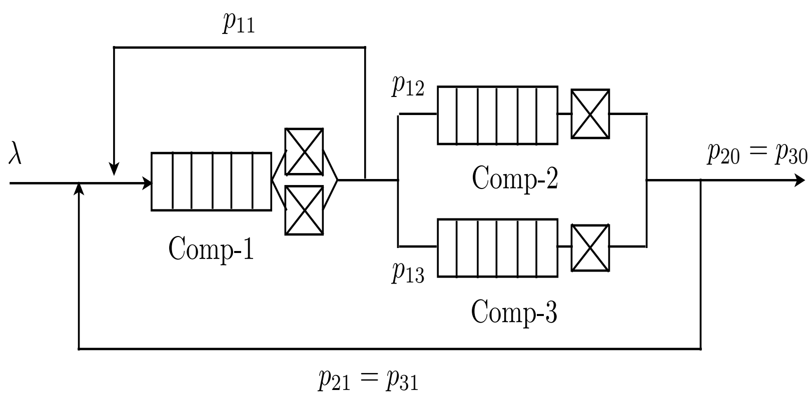
     - e.g. From the structure of the system (visit ratios in MVA and BCMP) or from synchronized measurements

  2) A request occupies a single component at a time

  - But arbitrary service disciplines and distributions or arrivals of requests

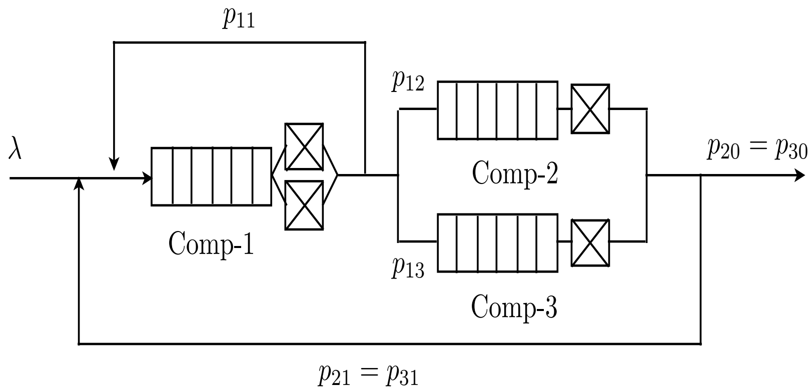
Successful case

- Centralized system architecture
  - Multiple servers
  - Coeff. of variation for service times: 0.5, 2 and 3
  - Probabilistic routing

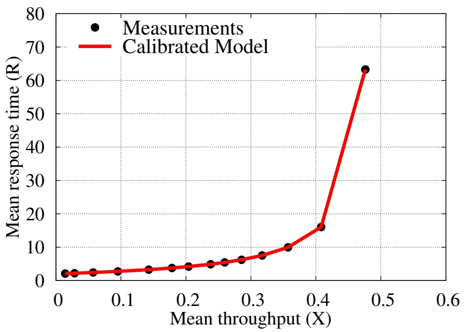


Successful case

- Centralized system architecture
  - Multiple servers
  - Coeff. of variation for service times: 0.5, 2 and 3
  - Probabilistic routing



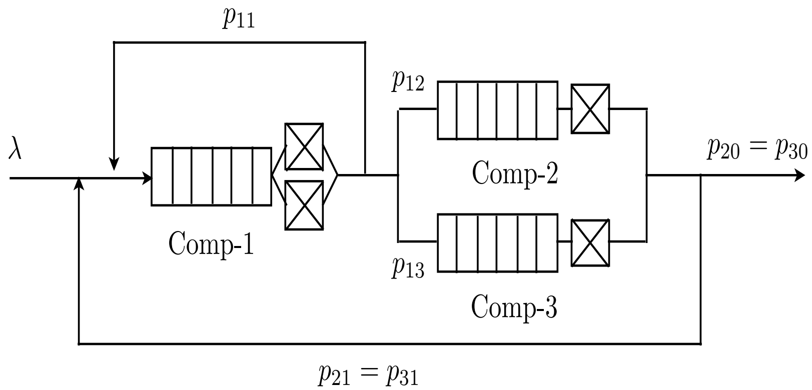
- **Calibrated individual component models**



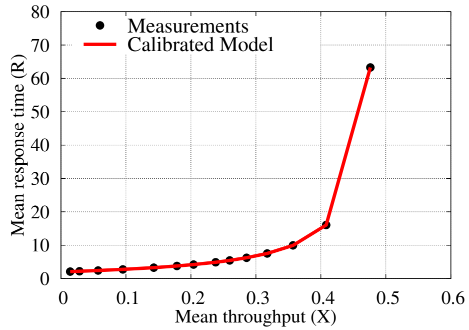
Component 1

Successful case

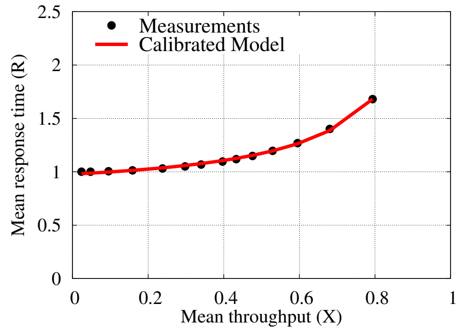
- Centralized system architecture
  - Multiple servers
  - Coeff. of variation for service times: 0.5, 2 and 3
  - Probabilistic routing



- **Calibrated individual component models**



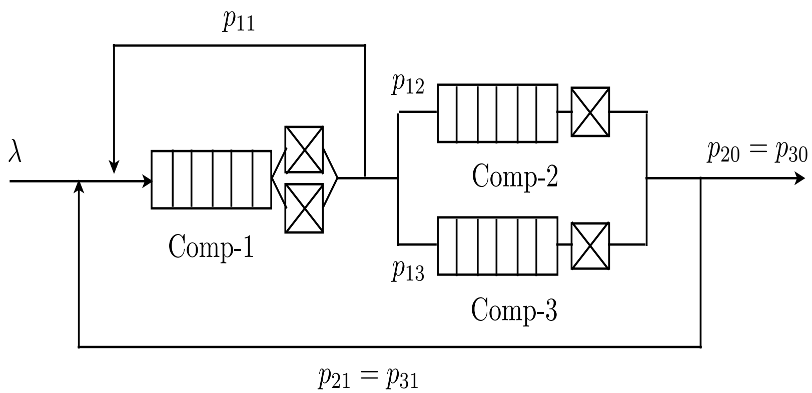
Component 1



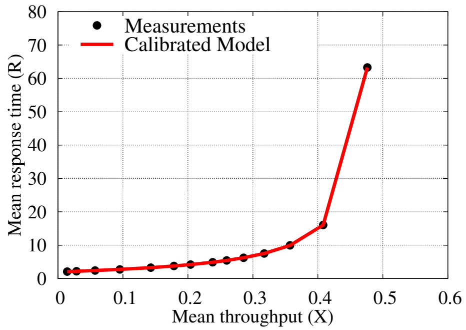
Component 2

Successful case

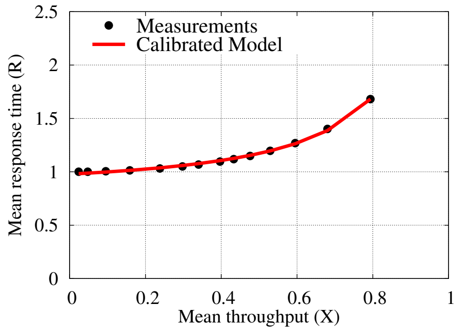
- Centralized system architecture
  - Multiple servers
  - Coeff. of variation for service times: 0.5, 2 and 3
  - Probabilistic routing



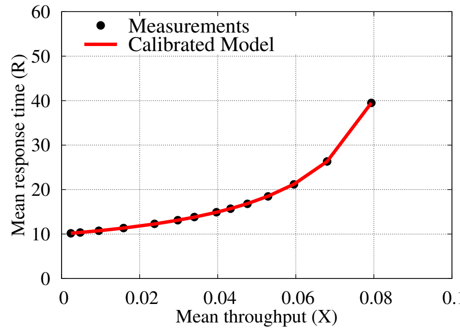
- **Calibrated individual component models**



Component 1



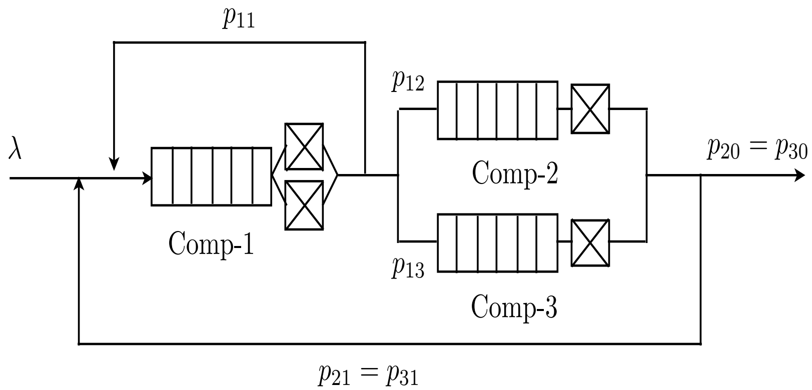
Component 2



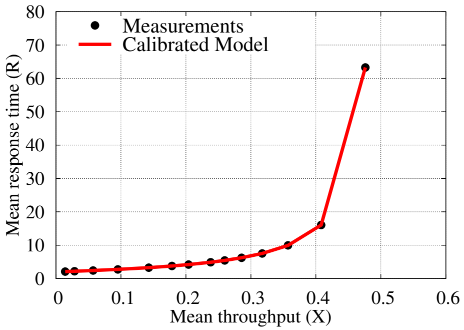
Component 3

Successful case

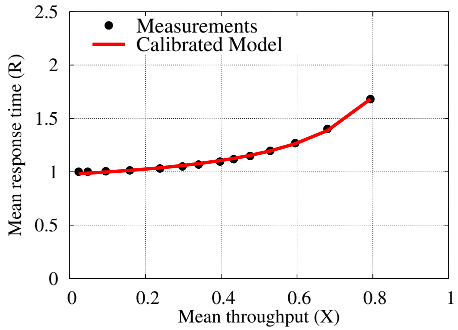
- Centralized system architecture
  - Multiple servers
  - Coeff. of variation for service times: 0.5, 2 and 3
  - Probabilistic routing



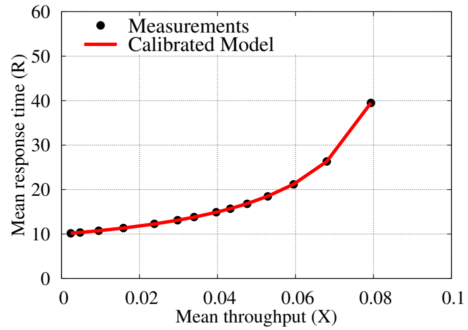
- **Calibrated individual component models**



Component 1



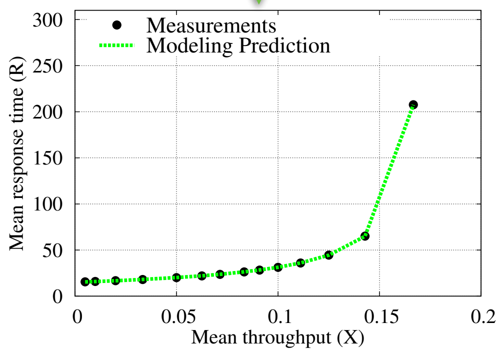
Component 2



Component 3

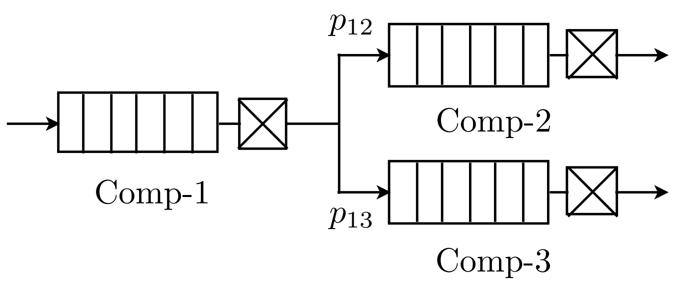
- **Combined overall model**

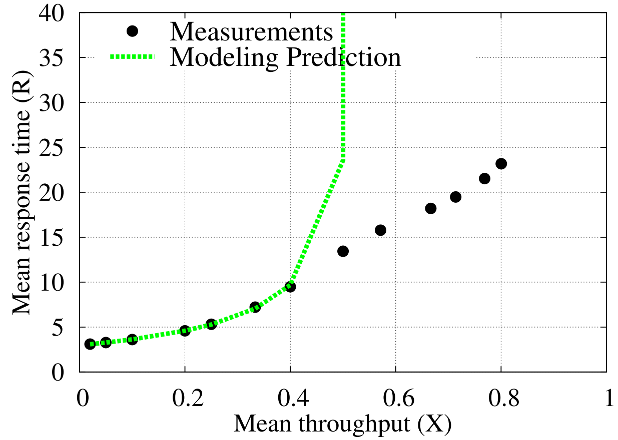
Prediction on the system response time



Cases of failure (1/2)

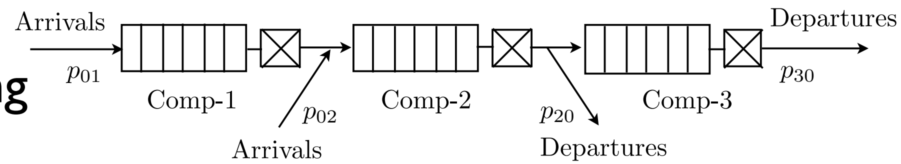
- State-dependent routing
  - Systems with load-balancing policies
  - e.g. IP networks, round robin DNS, cluster
    - If the current number of requests waiting in Comp-2 is smaller than 10,
      - Then requests are routed to Comp-2.
    - Otherwise, they are equally likely to be dispatched to Comp-2 and Comp-3.
  - **Throughput ratios are not constant**

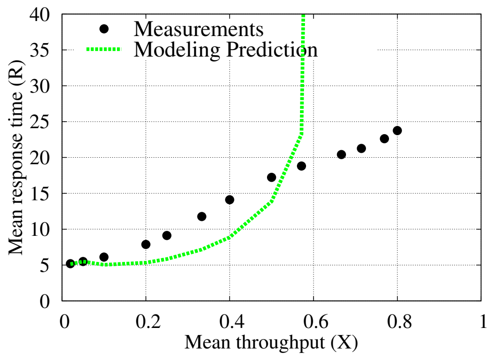




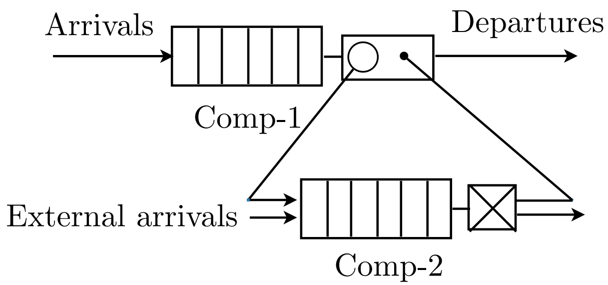
Cases of failure (2/2)

- Internal losses and arrivals
  - e.g. Due to buffer overflow, transmission errors, dynamic routing
  - Can be viewed as state-dependent routing
  - **Throughput ratios are not constant**



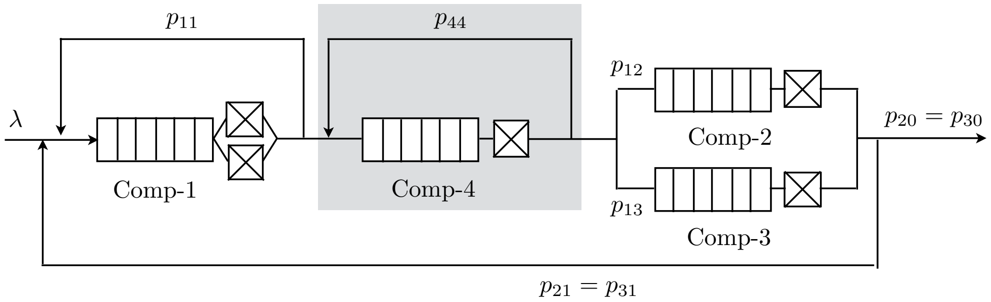


- Simultaneous resource possession
  - Requests may simultaneously "occupy" two or more resources
  - e.g. Databases and certain disk controllers
  - **Straightforward application of Little's law impossible**

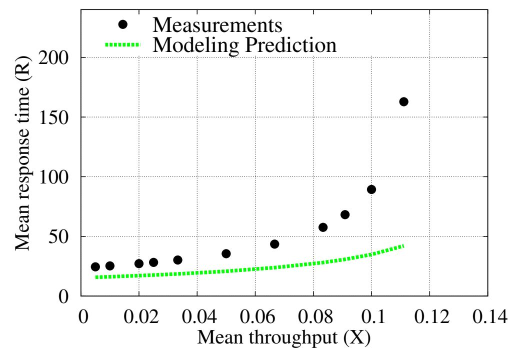


Discovery of an Unknown Component

- Centralized system architecture
  - All but one component have been instrumented, measured, and modeled
    - One component is unknown or neglected
    - e.g. Internal tables or buffers
- Our example
  - Measurements for Components 1, 2 and 3
  - No measurements for Component 4
    - Mean service time at component 4 is 10 times faster than at component 1
    - Deemed so fast that it is unlikely to be a factor in the overall system performance



Initial performance prediction



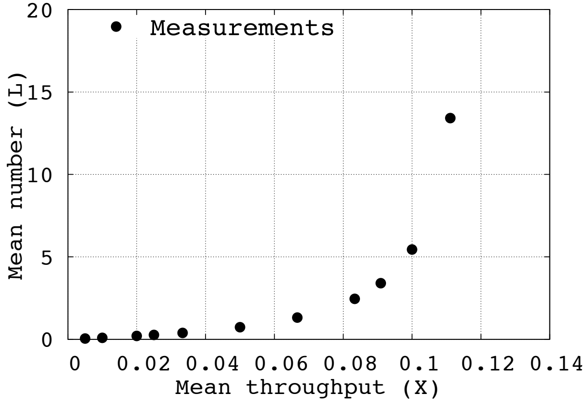
Clearly, the proposed approach is missing something!

Root Cause Analysis

Root Cause Analysis

- Difference in the performance between the system measurement points and the predicted performance
    - Observed error
    - Appears non-random

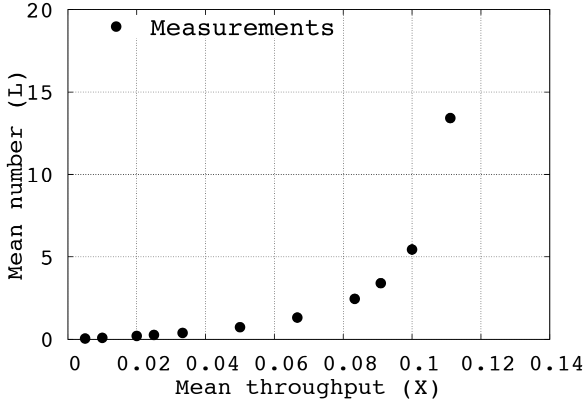
Residual performance



Root Cause Analysis

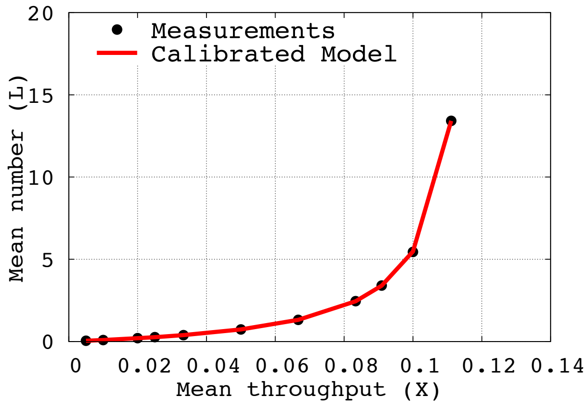
- Difference in the performance between the system measurement points and the predicted performance
  - Observed error
  - Appears non-random

Residual performance



- Fitting this residual behavior to a simple *M/M/1* queue

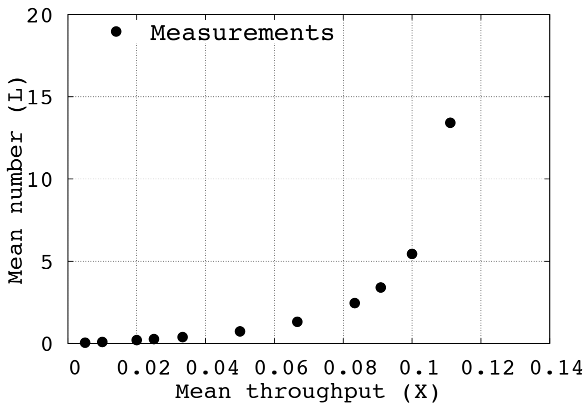
Good match!



Root Cause Analysis

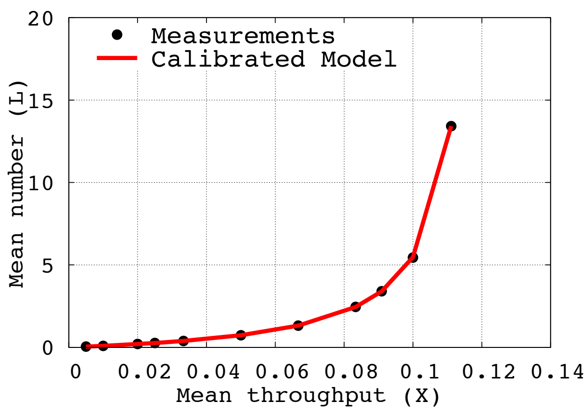
- Difference in the performance between the system measurement points and the predicted performance
  - Observed error
  - Appears non-random

Residual performance



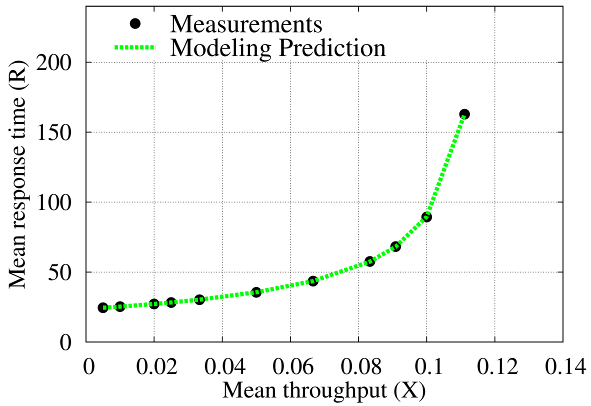
- Fitting this residual behavior to a simple *M/M/1* queue

Good match!



- It is likely that an additional component was not measured.
  - Adding the found *M/M/1* queue to the modeling approach improves overall match

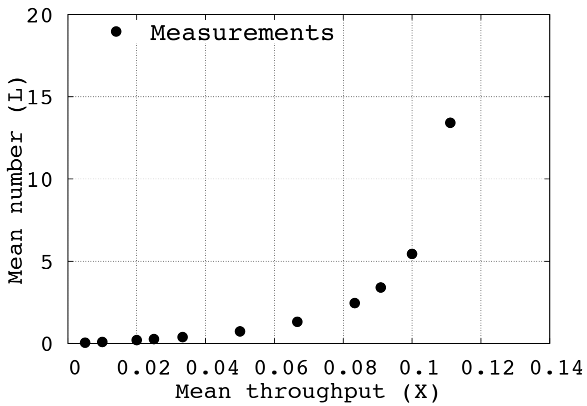
Refined performance prediction



Root Cause Analysis

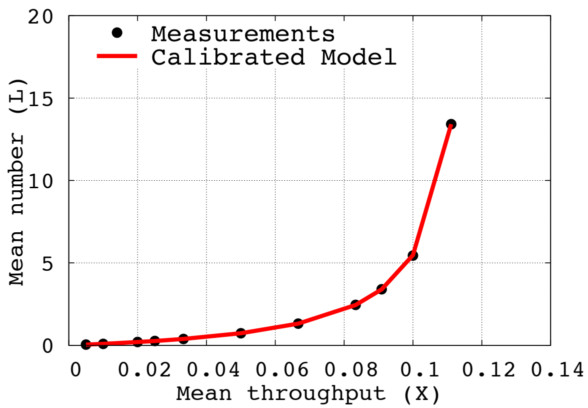
- Difference in the performance between the system measurement points and the predicted performance
  - Observed error
  - Appears non-random

Residual performance



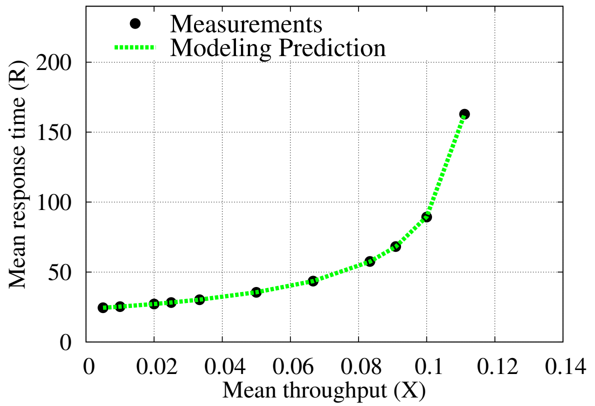
- Fitting this residual behavior to a simple *M/M/1* queue

Good match!



- It is likely that an additional component was not measured.
  - Adding the found *M/M/1* queue to the modeling approach improves overall match

Refined performance prediction



- If residual performance pattern more chaotic and not matched by a reasonable model
  - mere measurement "noise"
  - the system violates assumptions

Conclusions

- A simple approach for combining calibrated performance models of individual components into a system-level performance model
- Applicability conditions for open systems
- Analyzing the discrepancies between the model predictions and the measurements may be useful
- Future works:
  - distinguishing "measurement noise" from missing components
  - extending the approach to closed systems

References

[*Begin et al. 2010*] High-level Approach to Modeling of Observed System Behavior, T. Begin, A. Brandwajn, B. Baynat, B. Wolfinger, S. Fdida - Performance Evaluation, Volume 67, Issue 5.

Thank you!

Questions?