Performance Benchmark Fundamentals

Topics

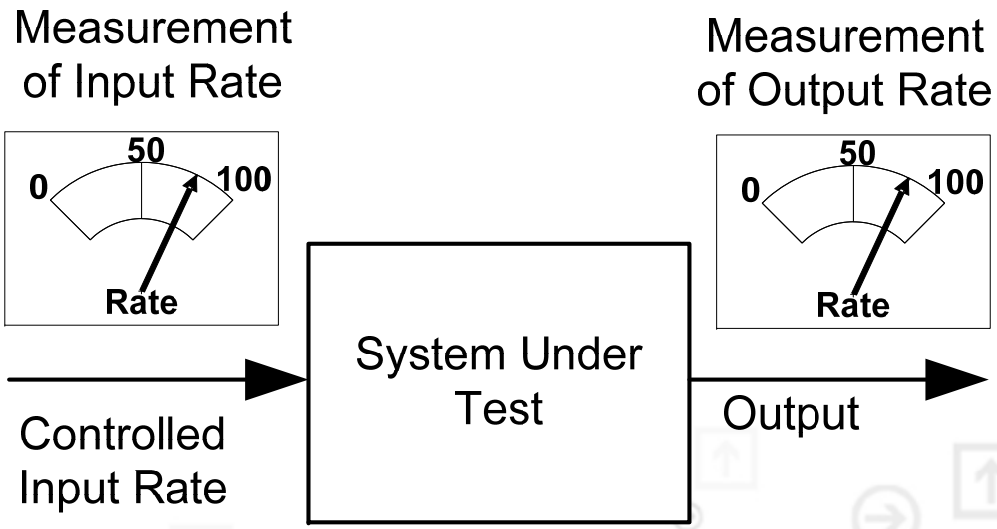
- Benchmark Basics
- Latency Measurements
- Resource Consumption Measurements
- Experimental Variables (Parameters)
- Test Harness Limitations
- Measurements with Multiple Components
- Measurements with Complex Components
- Interpreting Benchmarks
- Summary

Benchmark Basics

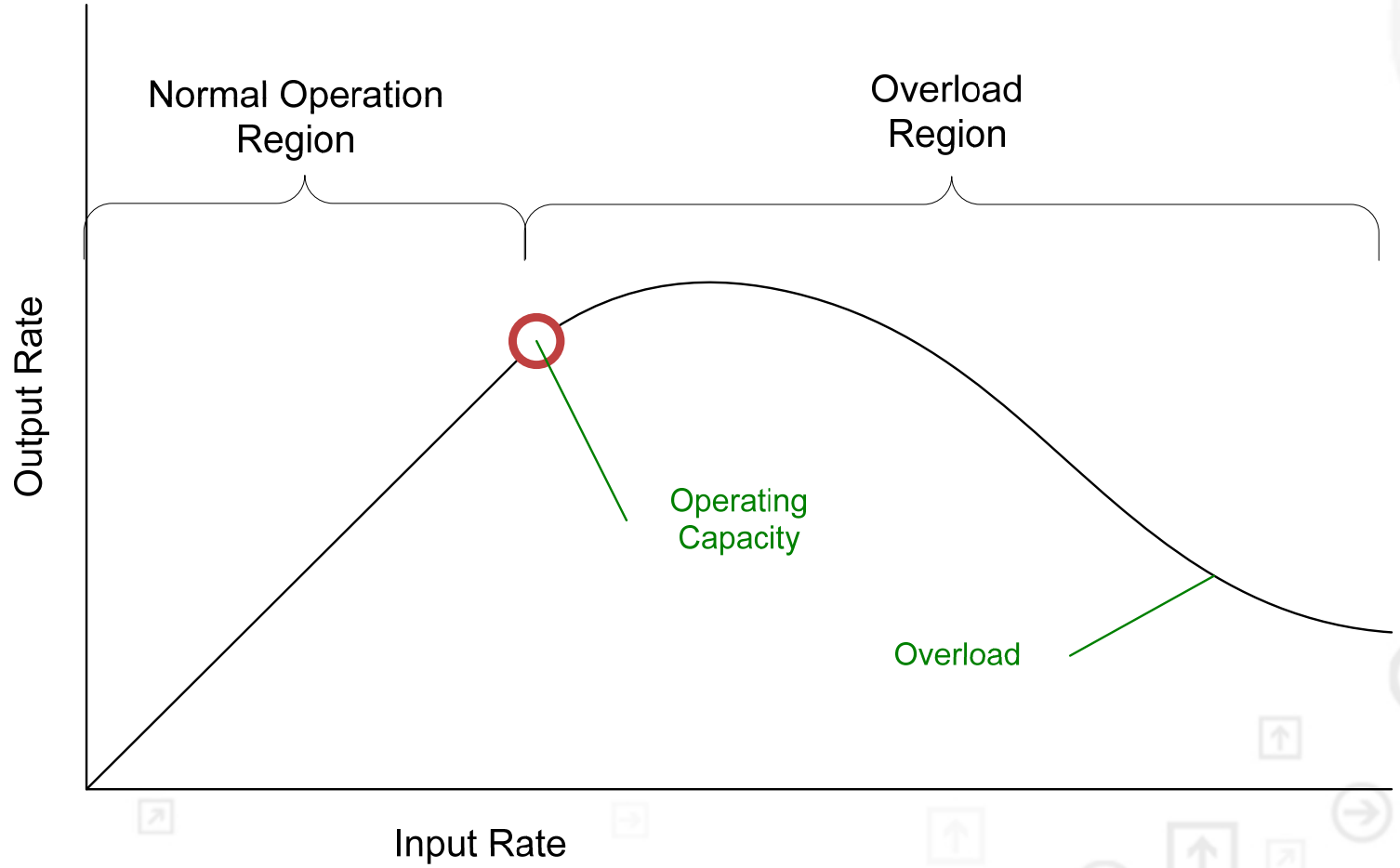
- **Benchmarks reflect the test environment**
  - You need to understand the environment in order to understand the benchmark
- **Measurements may reflect environment and test harness limitations, not necessarily system-under-test limitations**
- **The benchmark environment may not reflect your usage environment**

Typical Benchmark Experiment

- Control the input rate
- Wait for the system to stabilize
- Measure the output rate
- Repeat for different input rates

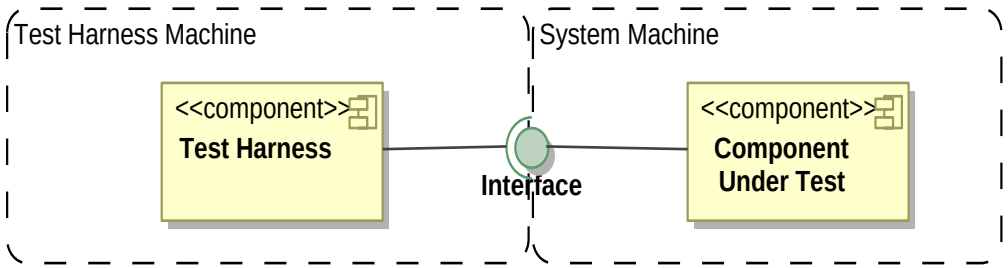


Typical Performance Curve

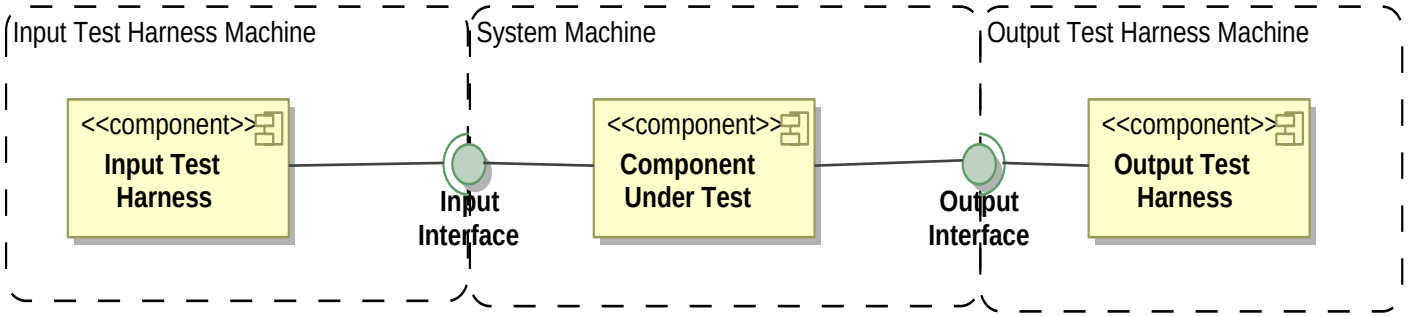


Document the Basic Experimental Design

- Components involved
- Interfaces used
- Placement on machines



a) Simple Request-Response Experimental Setup



b) Simple Straight-Through Processing Test Setup

Document the Experimental Details

System Under Test (repeat for each machine)

Processor	
CPU Type	x86
Number of CPUs	1
Cores/CPU	2
Clock Speed	2.66 GHZ
Memory	3GB
Network Interface Card	
Bandwidth (Mbit/second)	100
Half/Full Duplex	Full
Disk	
Number of Spindles	1
RPM	10,000
RAID Configuration	N/A
Access Bandwidth	10 MB/Sec
FT Ram Buffer?	No

Test Harness (repeat for each machine)

Processor	
CPU Type	x86
Number of CPUs	1
Cores/CPU	2
Clock Speed	2.66 GHZ
Memory	3GB
Network Interface Card	
Bandwidth (Mbit/second)	100
Half/Full Duplex	Full
Disk	
Number of Spindles	1
RPM	10,000
RAID Configuration	N/A
Access Bandwidth	10 MB/Sec
FT Ram Buffer?	No

Network

Backbone Bandwidth	1 Gbit

Experimental Data

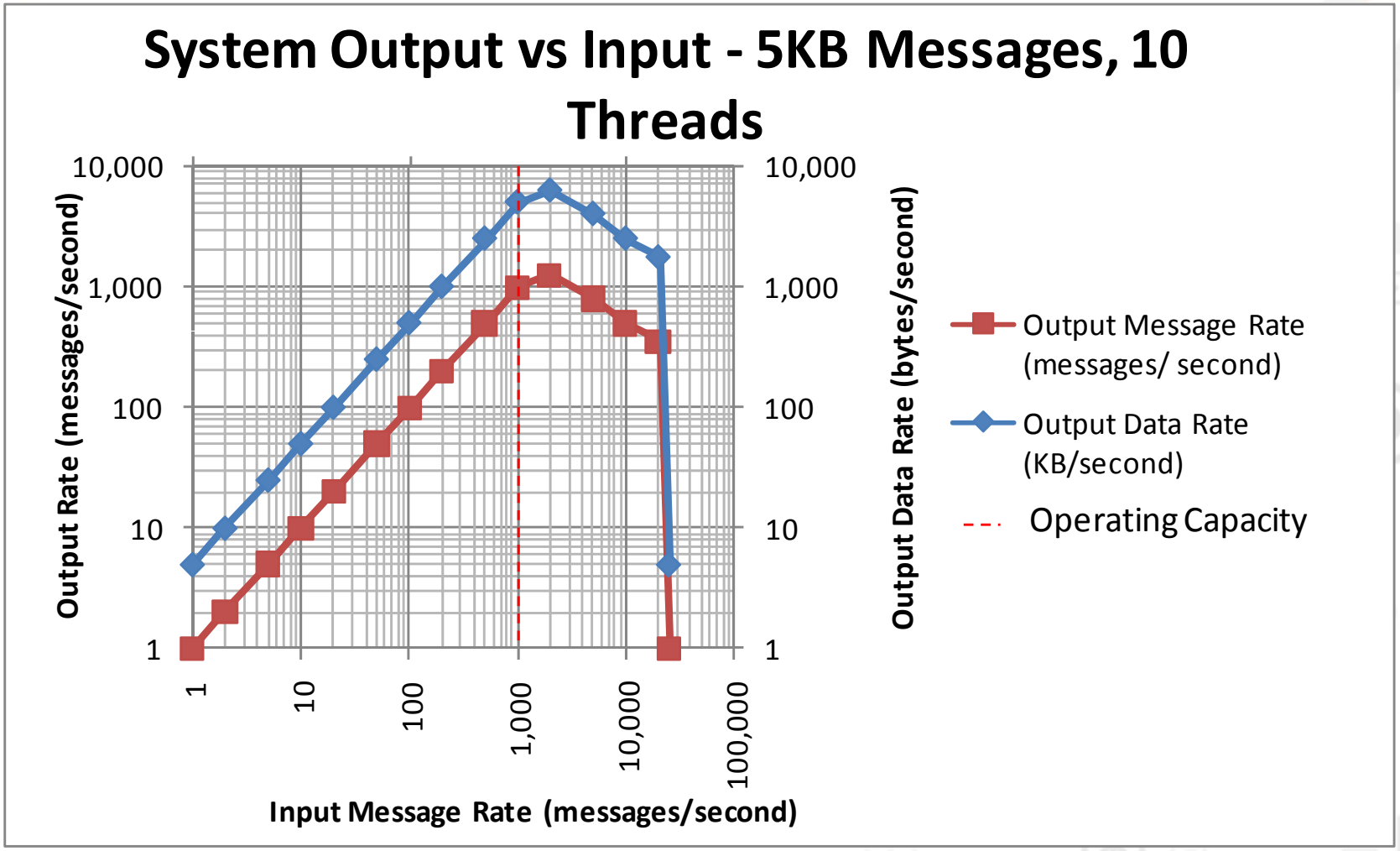
Experimental Paramters

Message Size (	5
Worker Thread	10

Test Data

Input Message Rate (messages/ second)	Input Data Rate (KB/ second)	Output Message Rate (messages / second)	Output Data Rate (KB/second)	Latency (milliseconds)	CPU Utilization (% of available CPU)	RAM Utilization (MBytes)	Disk Access Rate (accesses/ second)	Disk Bandwidth Utilization (MB/ second)	Network Bandwidth Utilization (KBytes/ second)	Available CPU
1	5	1	5	2	0%	150	3	0	10	100%
2	10	2	10	2	0%	150	6	0	20	100%
5	25	5	25	3	0%	150	15	0	50	100%
10	50	10	50	3	1%	151	30	0	100	100%
20	100	20	100	4	1%	151	60	0	200	100%
50	250	50	250	4	3%	153	150	1	500	100%
100	500	100	500	5	5%	155	300	2	1,000	100%
200	1,000	200	1,000	5	10%	160	600	3	2,000	100%
500	2,500	500	2,500	6	25%	175	1,500	8	5,000	100%
1,000	5,000	1,000	5,000	6	50%	200	3,000	15	10,000	100%
2,000	10,000	1,250	6,250	10	100%	250	3,750	19	16,250	100%
5,000	25,000	800	4,000	38	100%	400	2,400	12	29,000	100%
10,000	50,000	500	2,500	120	100%	650	1,500	8	52,500	100%
20,000	100,000	350	1,750	343	100%	1,150	1,050	5	101,750	100%
25,000	125,000	1	5	150,000	100%	1,400	3	0	125,005	100%

Basic Benchmark Results



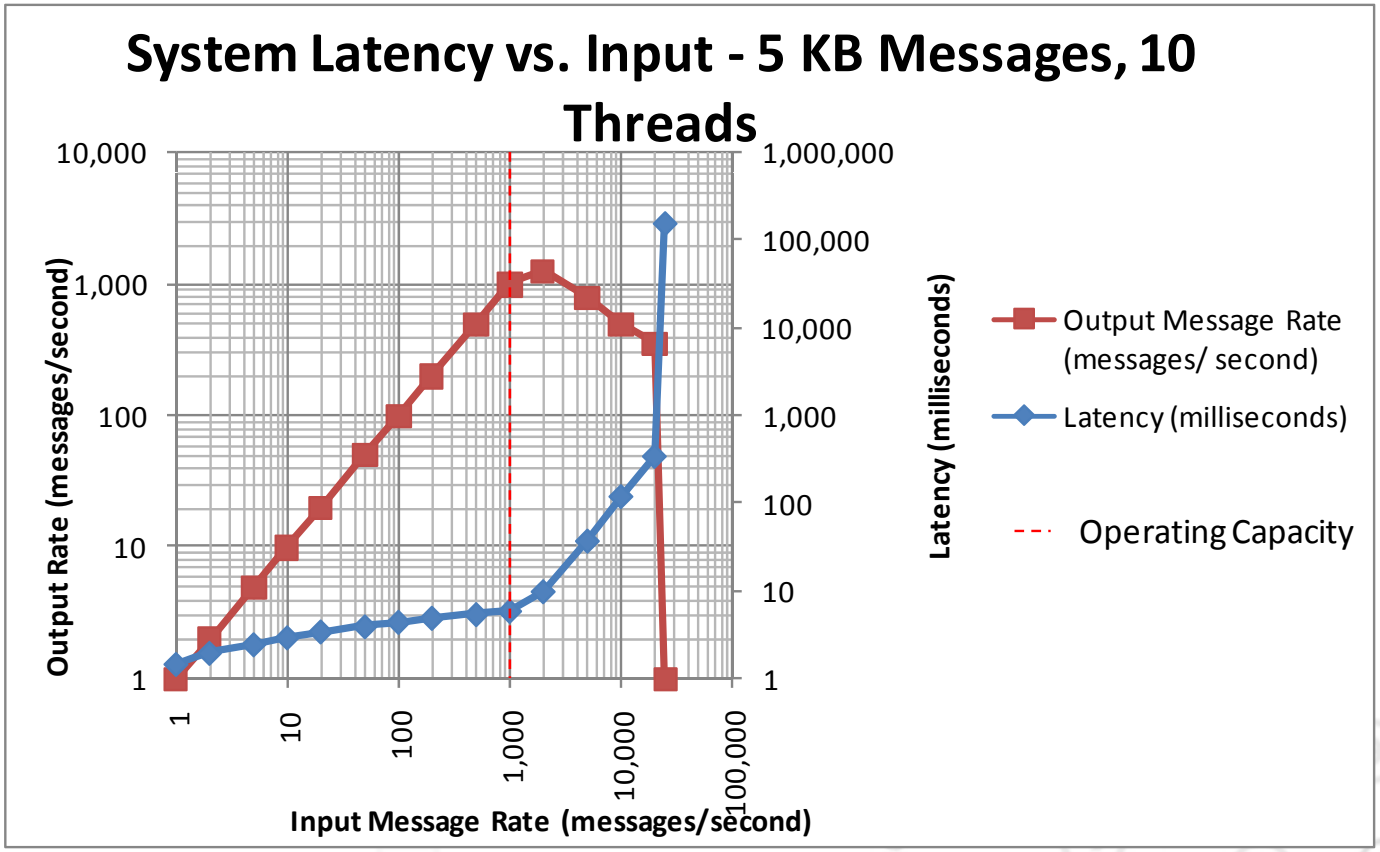
Potentially Misleading Experiments

- **Overload Tests (drain-the-bathtub)**
  - Inputs provided at an extremely high rate
  - Measure time required to generate all resulting outputs
    - outputs/time = rate (most likely in overload range!)

- **Low-End Performance Tests**
  - Data points well below operating capacity
  - Yield no insight as to what the operating capacity is
    - Cannot be used to compare two designs

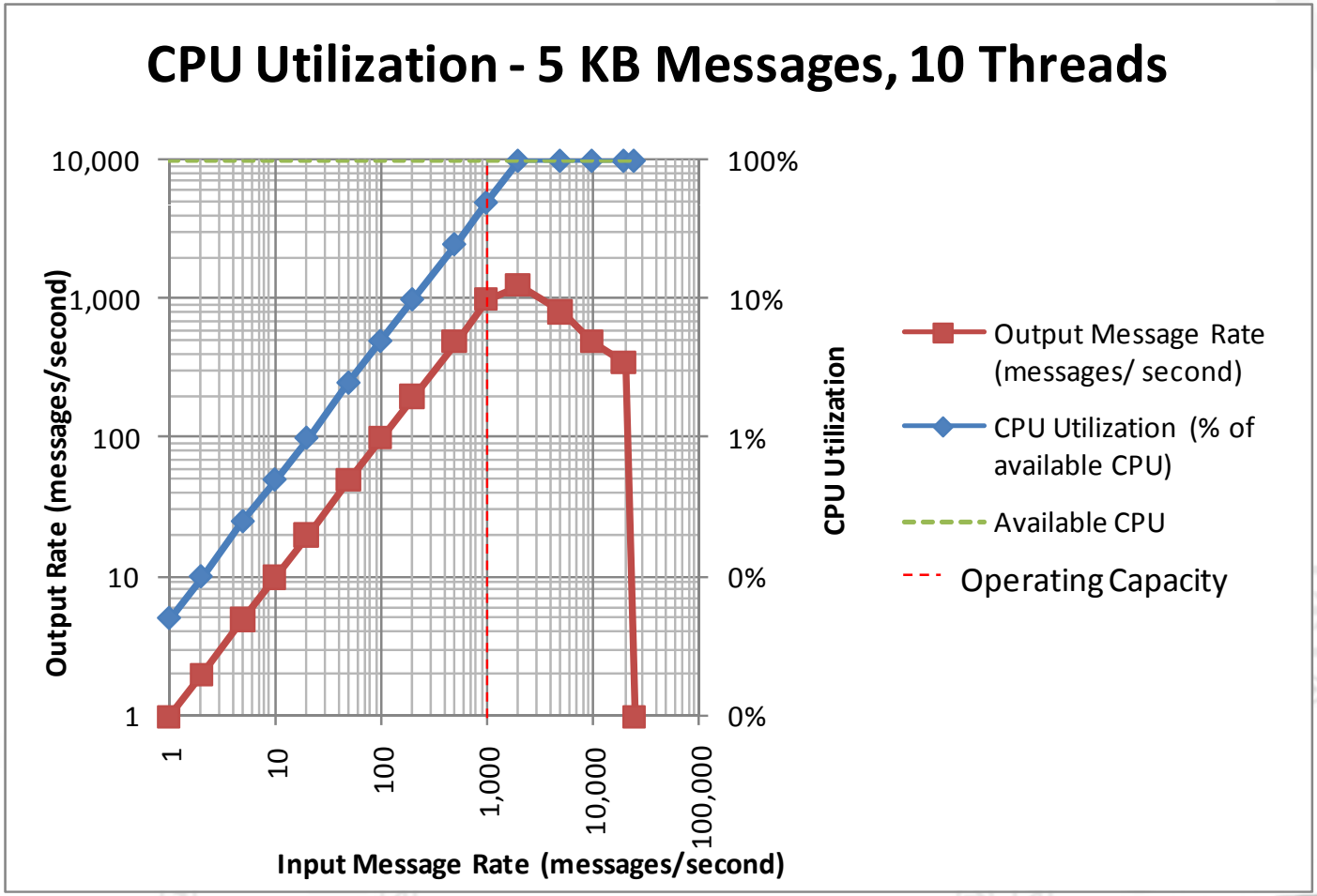
Latency Measurements

- Usually change markedly at operating capacity
  - Good idea to show basic capacity graph as well



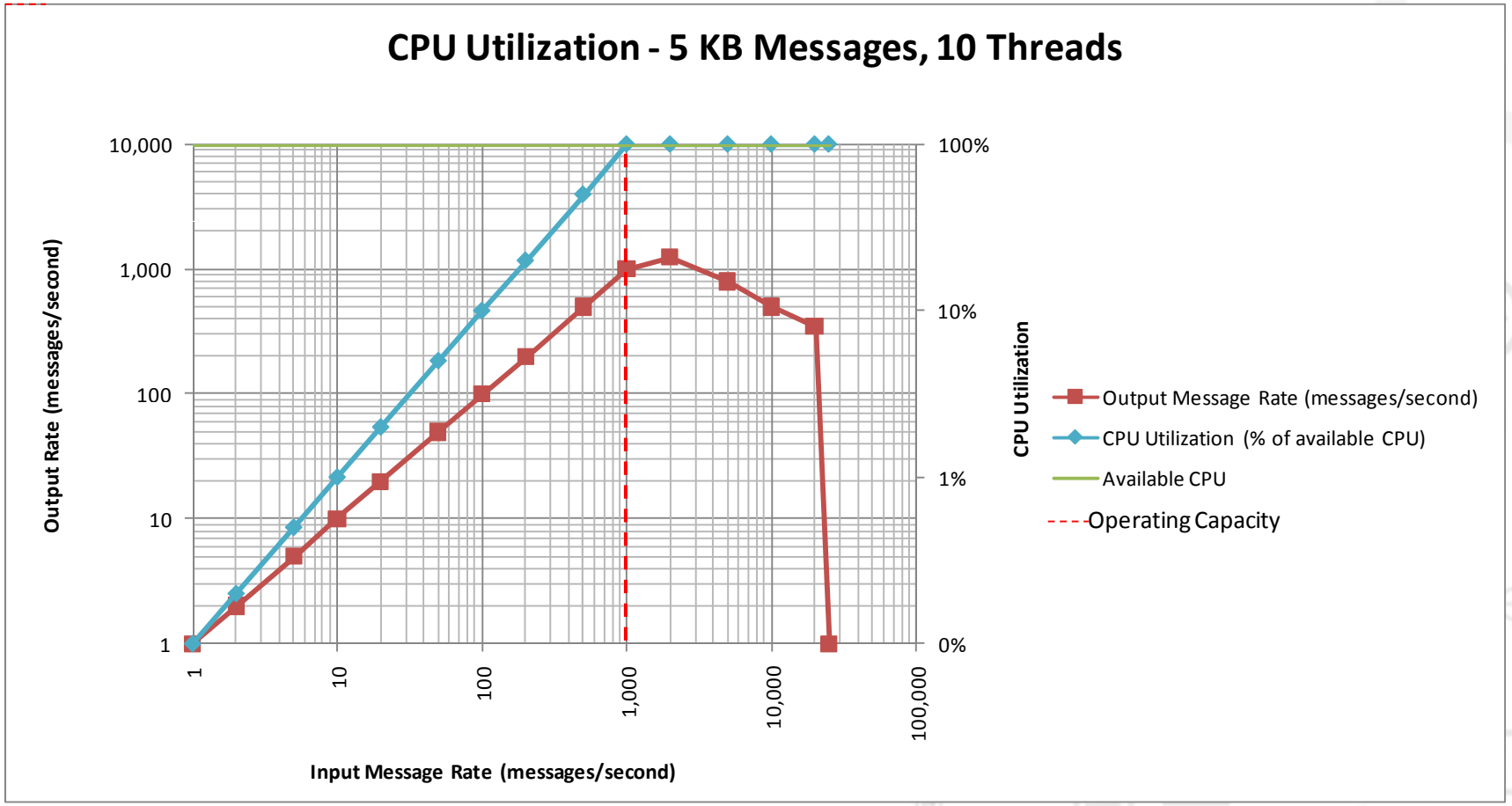
CPU Utilization – CPU Not Limiting Performance

- CPU utilization peaks ***after*** operating capacity is reached



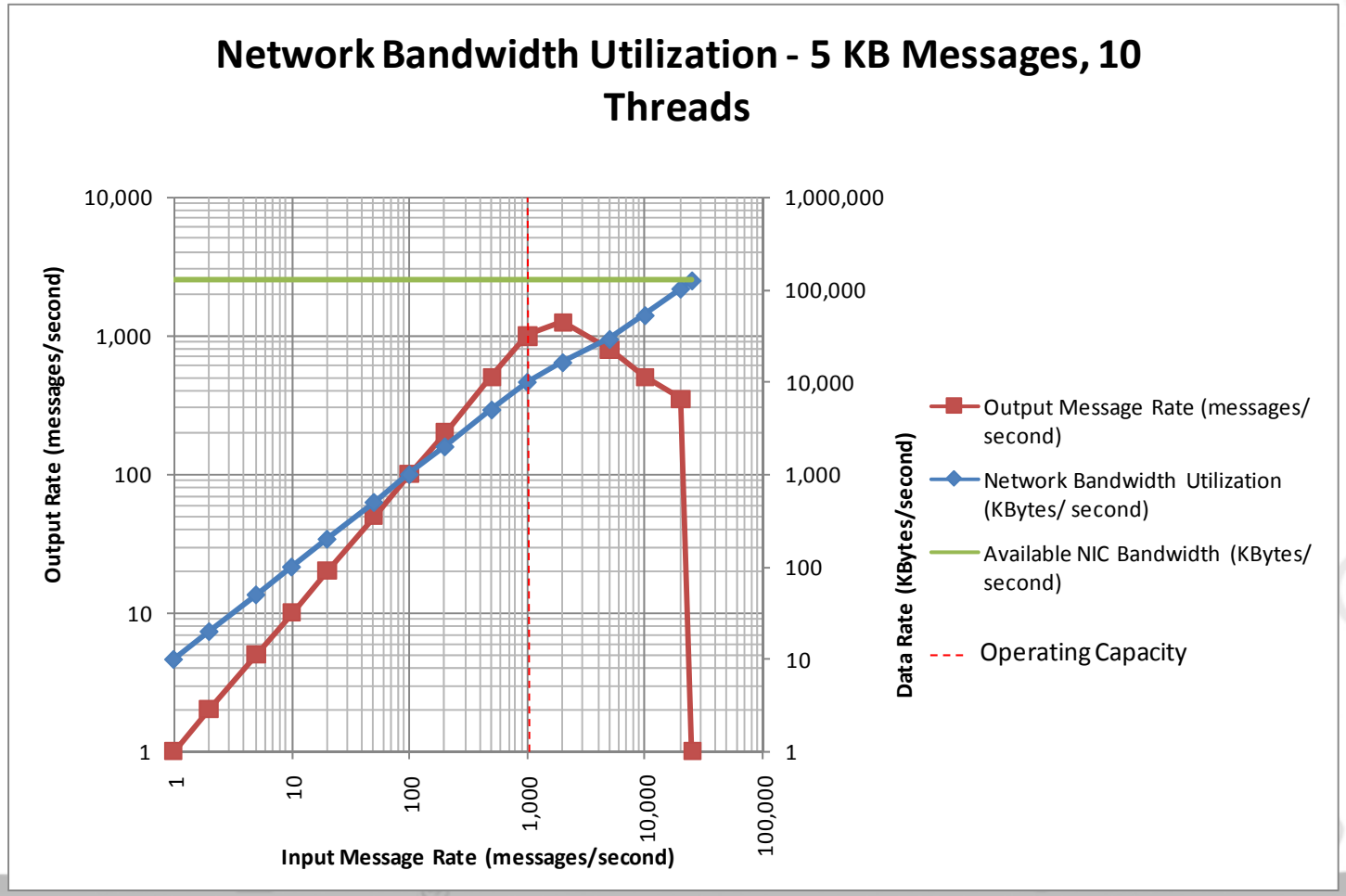
CPU Utilization – CPU Limiting Performance

- CPU Utilization Peaks *At* Operating Capacity



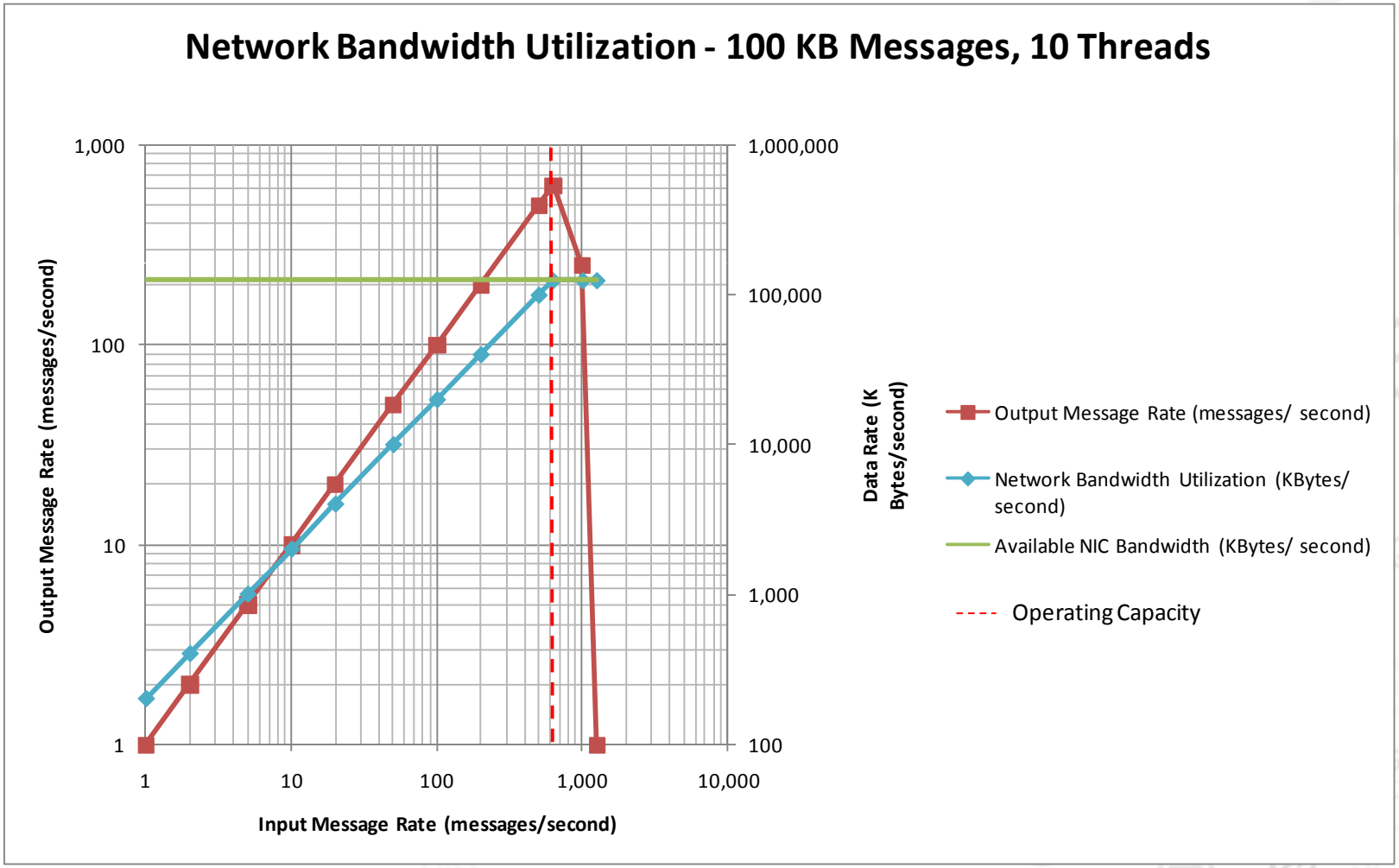
Network Bandwidth – Network Not Limiting Performance

- Network peaks ***after*** operating capacity is reached
  - May not peak at all



Network Bandwidth – Network Limiting Performance

- Network peaks *at* operating capacity



Disk Utilization is Complex to Evaluate

- **Two different limitations:**
  - Disk access rate - # of accesses/second
    - Comes into play when data blocks are small to modest in size
  - Data transfer rate - # of bytes/second
    - Comes into play when large data blocks are being written/accessed
- **Application's use of disk is also a factor**
  - Read vs. Write
  - Synchronous vs. Asynchronous
    - Critical writes should ***always*** be synchronous

Key Disk Performance-Related Questions

- Is the disk part of the machine using it, or is it in a separate storage subsystem?
- If it is in a storage subsystem, does the subsystem have a fault tolerant buffer?
- Is the disk a single physical disk, or a RAID array? If it is a raid array, how many spindles are in the array, and what RAID configuration is being used?
- What is the rotational speed of each disk?
- What is the average seek time of the disk head?
- What is average data transfer rate for the disk?
- What is the size of the buffer cache for the disk?

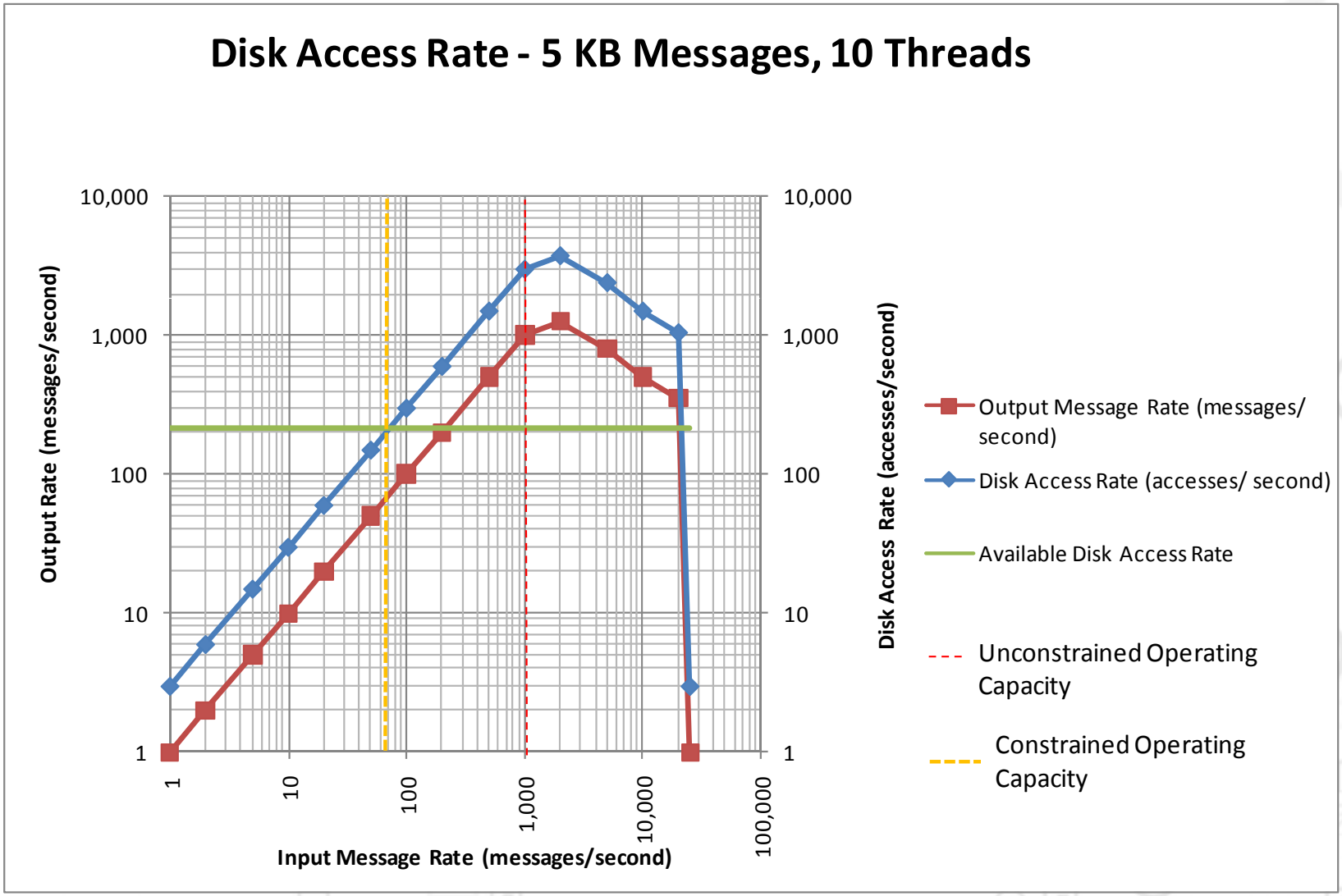
Disk Formulas (assuming no fault-tolerant buffer)

$$avgRotationalLatency_{\text{sec}} = \frac{1}{(diskSpeed_{RPM} / 60) \times 2} = \frac{1}{diskSpeed_{RPM} / 30}$$

$$dataTransferTime_{\text{sec}} = \frac{messageSize_{bytes}}{transferRate_{bytes/\text{sec}}}$$

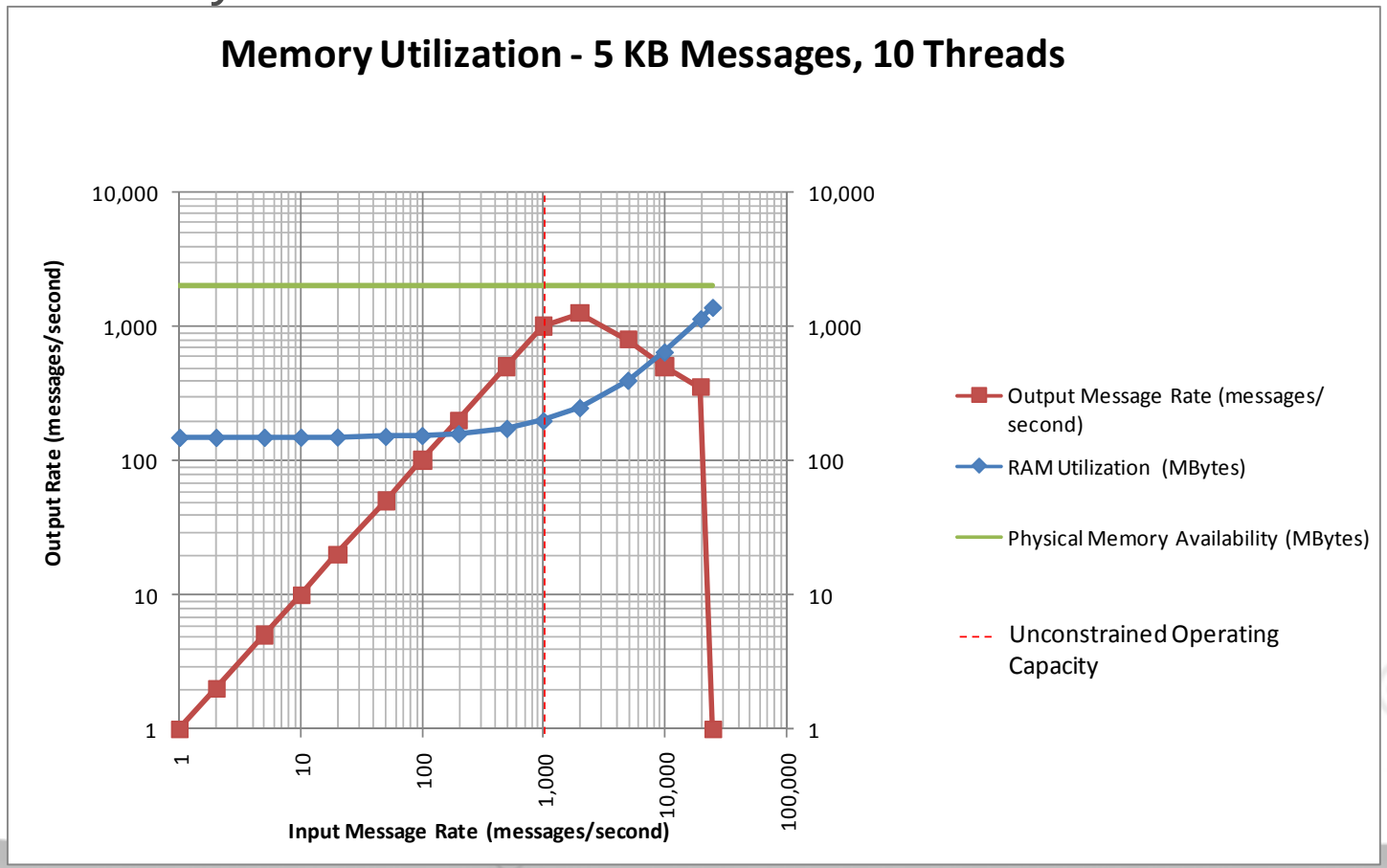
$$avgAccessRate_{\text{sec}} = \frac{1}{\max\left(avgSeekTime + avgRotationalLatency\right) + dataTransferTime}$$

Impact of Disk on Performance



Memory Utilization – Memory Not a Limiting Factor

- Memory utilization peaks ***after*** operating capacity reached
  - May not hit limit at all



Experimental Variables (Parameters)

- **The variables that are important depend upon the nature of the components and the nature of the tests**
- **Typical parameters:**
  - The number of threads in each system component
  - The sizes of buffers
  - Constraints on the number of activities that can occur in parallel
  - The number of allowed connections (there may be different numbers for different connection types)
  - The number of concurrent sessions
- **Don’t forget the test harness parameters!**

Test Harness Limitations

If

you reach an apparent capacity limit

and

you can't identify system resources that have reached limits

then either

a) there is a system design characteristic that limits its ability to utilize the available resources

e.g. insufficient threads

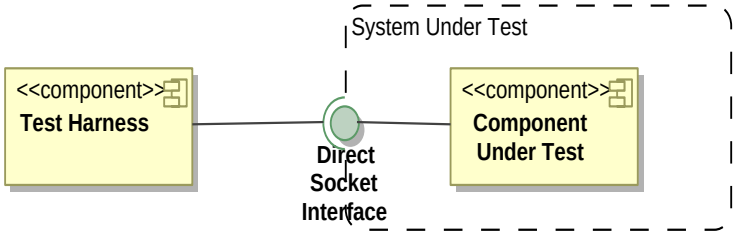
or

b) The test harness is incapable of driving the system to its full capacity

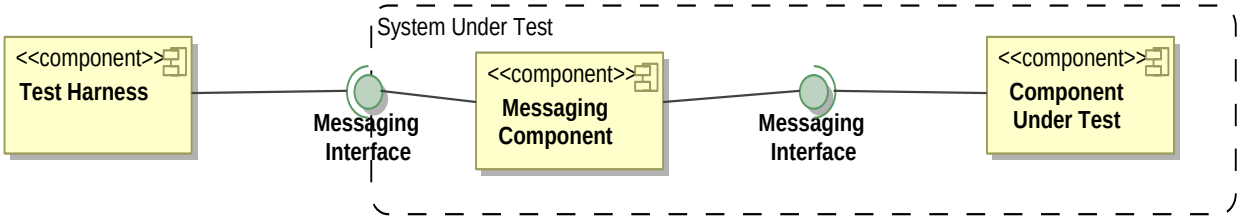
Make sure you know what the real limitation is!

Multi-Component Tests

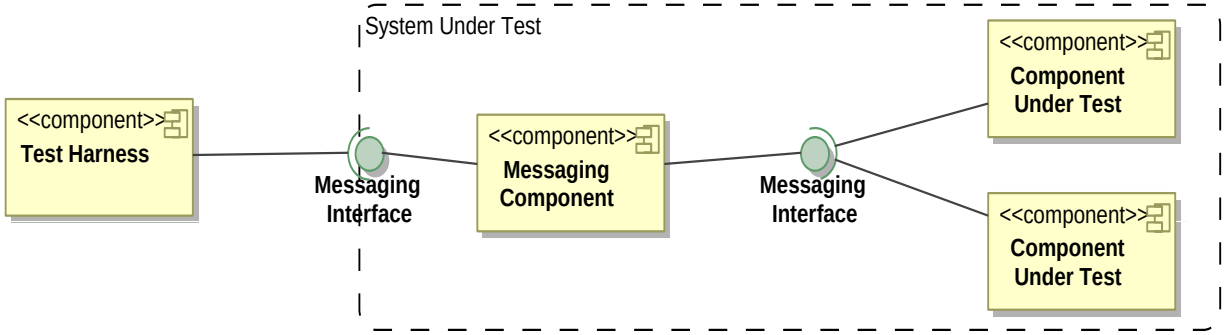
- Comparing different designs can be tricky



a) Direct Socket Interaction



b) Messaging Interaction



c) Messaging Interaction with Load Distribution

Comparing Designs with Different Communications

- **Adding a messaging component introduces latency**
- **If the test harness has a fixed number of threads, each waiting for a reply, *the addition of latency will reduce the number of requests/second that the harness is capable of delivering***
  - An apparent reduction in throughput will result
- **A meaningful comparison requires that the test harness be able to fully load both designs**

Comparing Designs with and without Load Distribution

- **Load distribution requires a distribution mechanism**
  - e.g. JMS queue
- **Load distribution mechanism will both add latency (see previous section) and consume additional resources**
- **A LB configuration with only one worker will consume more resources than the non-LB solution**
- **You won't see the LB benefits until you add a second worker**
  - You may need to place different components on different machines as well

Complex Components

- **Typically perform more than one task**
- **Different tasks consume varying levels of resource**
  - e.g. iProcess
- **Testing strategy (simplistic)**
  1. Identify simplest (lightest-weight) tasks and generate baseline performance for these tasks
     - E.g. start a trivial workflow, execute a trivial task
     - These measurements basically measure raw overhead
     - Adding real work will *always degrade performance*
  2. Vary parameters for baseline tests
     - i.e. input size, threads, etc.
  3. Measure performance for individual non-trivial tasks
     - Vary parameters, compare against baseline
     - Gives you incremental cost for the non-trivial task
  4. Explore scenarios in which tasks interact
     - Data for individual tasks no longer predicts overall performance
     - Pragmatics will limit the number of scenarios that can be explored – make them representative of real usage

Interpreting Benchmarks as Guides to Your Design

- **Contention for resources**
  - What else is competing for the resources in your design?
- **Variable resource availability**
  - Difficult to use networks to capacity in production environments
    - Benchmark may be predictive up to 30% of resource availability, inaccurate at higher levels
- **Extrapolation of results**
  - Benchmark environment may be different
    - Can you accurately extrapolate?
      - Windows->Linux?
      - Intel->RISC?
      - Laptop->Server?
- **When in doubt, do your own benchmark**
  - This is why it is important to understand how the benchmark was performed

Summary

- **Always document the test design in sufficient detail to allow others to accurately reproduce your results.**
- **Always range demand until the operating capacity of the system under test has been reached (i.e. further increases in input rate do not result in proportional increases in output rate)**
- **Always document measured or estimated resource availability and consumption**
- **Once an apparent operational limit has been reached, investigate to determine whether a true resource constraint has been reached. Consider adding resources (i.e. adding memory, changing to a higher network bandwidth).**
- **If an apparent operational limit has been reached without exhausting available resources:**
  - Consider whether tuning the system under test might further increase the operational capacity.
  - Consider whether the design or configuration of the test harness might be the true limiting factor in the experiment.