Performance

rambo

2014.7

- What is Performance?

performance

- a measure that relates time to the executions of individual tasks
- Performance is an attribute of each individual experience with a system.

time

- the indefinite continued progress of existence and events in the past, present, and future regarded as a whole: "Time is what prevents everything from happening at once."
—John Archibald Wheeler (1911-2008)

task

- a piece of work to be done or undertaken

- A **task** is a business unit of work, named and described in the language of the business.

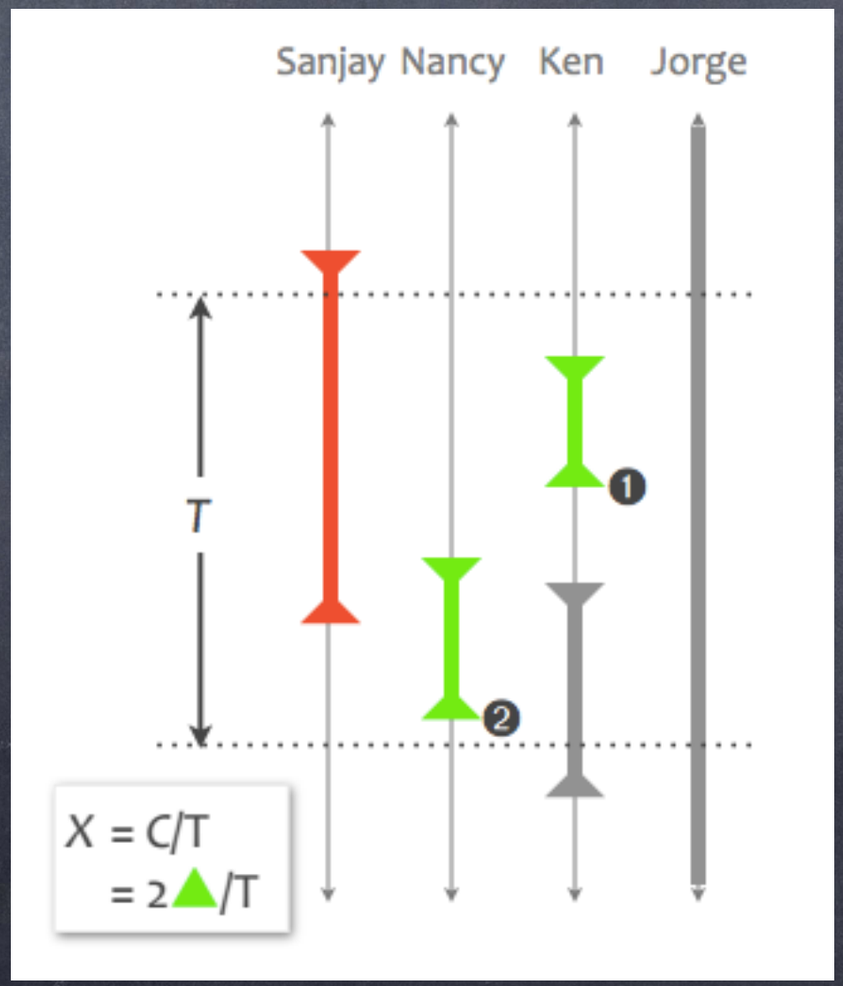
experience

- an execution of a task.

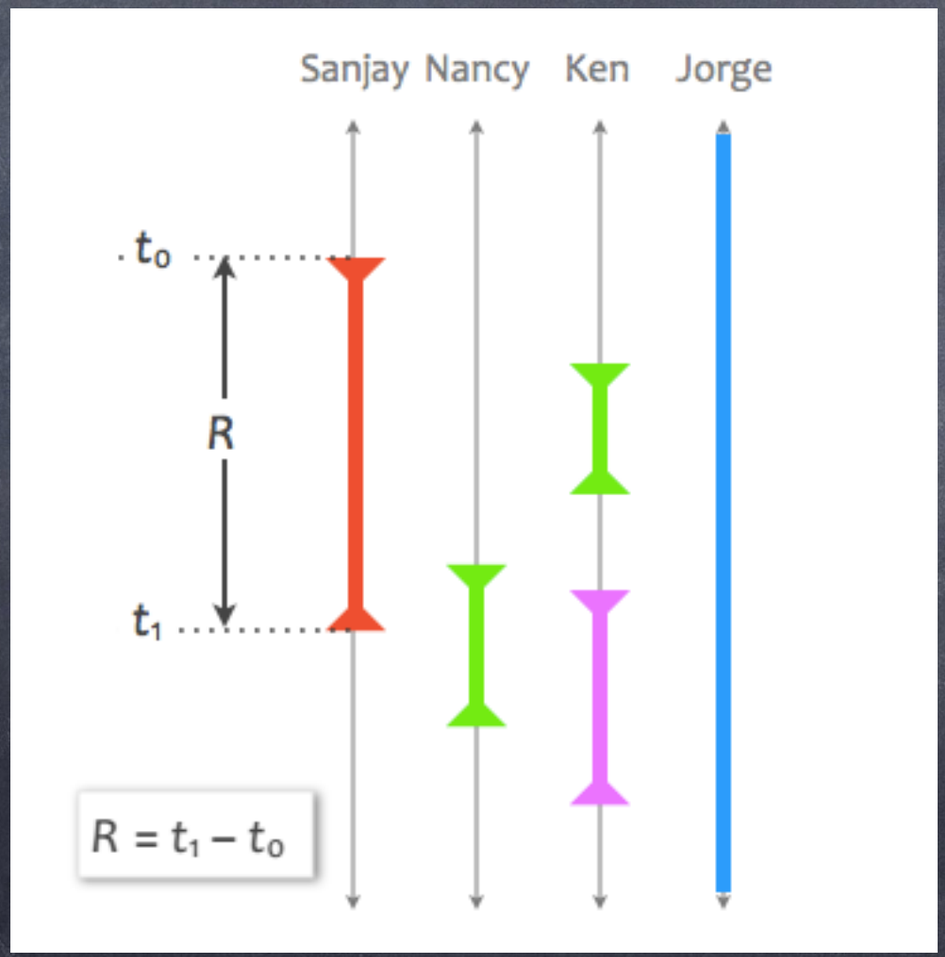
- Two ways to relate experiences to time
- experiences/time
- time/experience

- throughput
  - output or production, as of a computer program, over a period of time
- response time
  - the duration taken for a system to react to a given stimulus or event

- throughput (X) = experiences/time



- response time (R) = time/experiences



- Throughput is important to groups, leaders.
- Response time is important to individuals, leaders.

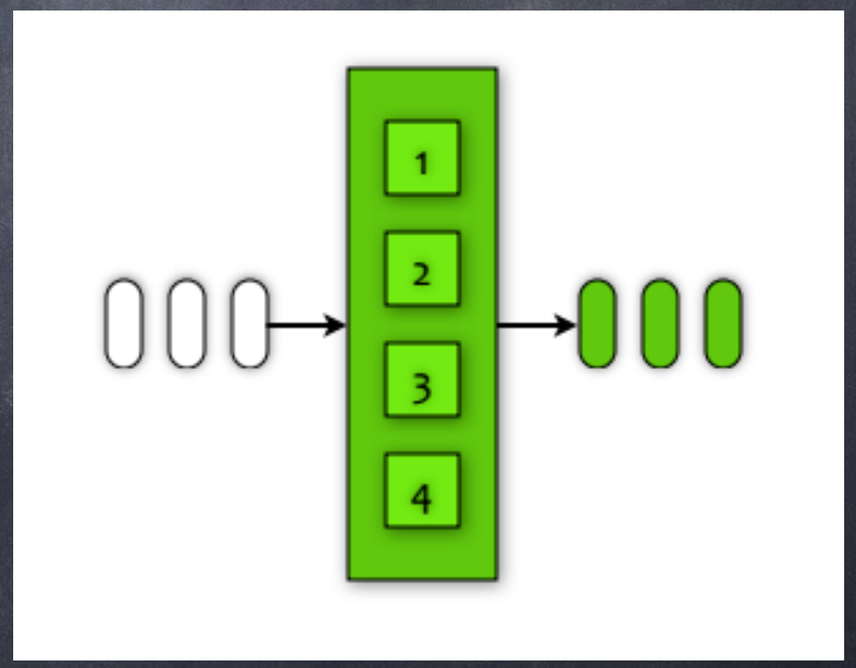
$$X(N) = \frac{N}{R(N) - Z}$$

- average throughput
- average response time

$$R(N) = \frac{N}{X(N)} + Z$$

- Case:收费站
- N:通道数, Z:常量(waiting time)

- N=4, Z=0
- if R(N)=1, then X(N)=4
- if X(N)=8, then R(N)=0.5



Percentile Specifications

- all response must be in <= 1.0s

Case: 1-second tolerance

- which response times do you like better
- R(N)= 1.0s

	R(N)
A	**1.000 s**
B	**1.000 s**

	List A	List B
1	.924	.796
2	.928	.798
3	.954	.802
4	.957	.823
5	.961	.919
6	.965	.977
7	.972	1.076
8	.979	1.216
9	.987	1.273
10	1.373	1.320

Case: 1-second tolerance

- which response times do you like better

	R(N)	Success rate
A	1.000 s	**90%**
B	1.000 s	**60%**

	List A	List B
1	.924	.796
2	.928	.798
3	.954	.802
4	.957	.823
5	.961	.919
6	.965	.977
7	.972	1.076
8	.979	1.216
9	.987	1.273
10	1.373	1.320

- all response must be in <= 1.0s for >= 90% of executions
- users feel the variance, not the mean

Case: 1-second tolerance

- which response times do you like better

	R(N)	Success rate
A	1.000 s	**90%**
C	1.000 s	**90%**

	List A	List C
1	.924	.091
2	.928	.109
3	.954	.134
4	.957	.136
5	.961	.159
6	.965	.172
7	.972	.185
8	.979	.191
9	.987	.207
10	1.373	8.616

- all response must be in <= 1.0s for >= 90% of executions, all <= 5.0s for >= 99% of executions

Case: 1-second tolerance

- which response times do you like better

	R(N)	1-sec tolerance success rate	**5-sec tolerance success rate**
A	1.000 s	90%	**99%**
C	1.000 s	90%	**90%**

	List A	List C
1	.924	.091
2	.928	.109
3	.954	.134
4	.957	.136
5	.961	.159
6	.965	.172
7	.972	.185
8	.979	.191
9	.987	.207
10	1.373	8.616

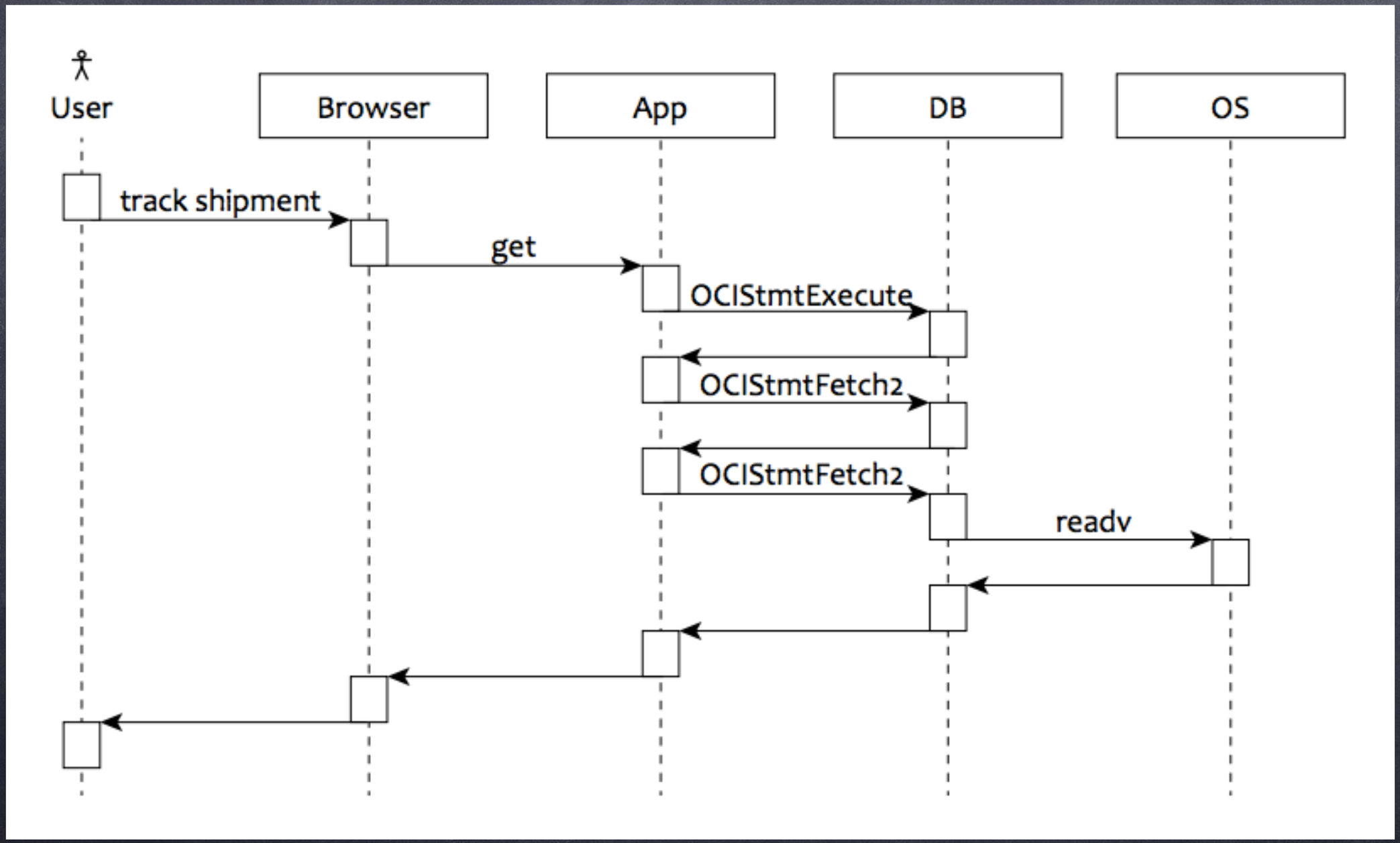
Problem Diagnosis

- the right **start** is the **most important** thing

- What is the current state?
- What is the goal state?

- what is the goal state is impossible?
- How can you know?

The Sequence Diagram



sequence diagram

- good conceptual tool
- but doesn't scale

The Profile

- a tabular account of response time, in which the sum of component response times exactly equals the total response time being measured

The Profile

CALL-NAME	DURATION	%	CALLS	MEAN
db file sequential read	59,081.406102	76.6%	10,013,394	0.005900
log buffer space	6,308.758563	8.2%	9,476	0.665762
free buffer waits	4,688.730190	6.1%	200,198	0.023420
EXEC	4,214.190000	5.5%	36,987	0.113937
log file switch completion	1,552.471890	2.0%	1,853	0.837815
db file parallel read	464.976815	0.6%	7,641	0.060853
log file switch (checkpoint incomplete)	316.968886	0.4%	351	0.903045
rdbms ipc reply	244.937910	0.3%	2,737	0.089491
undo segment extension	140.267429	0.2%	1,411	0.099410
log file switch (private strand flush incomplete)	112.680587	0.1%	134	0.840900
17 others	23.367228	0.0%	58,126	0.000402
TOTAL (27)	77,148.755600	100.0%	10,332,308	0.007467

The Profile

- you've done this before if you ever used...
- java -prof ...; java ProfileViewer ...
- gcc -pg ...; gprof ...

- Where did my code spend my time?
- How long should this task run?

answer

- what is the goal state is impossible?
- How can you know?
- Profile is how you can know.

Bottleneck

- the resource that dominates a given task's response time

- bottleneck = profile's top line

- Quiz: What's the task's bottleneck?

CALL-NAME	DURATION	%	CALLS	MEAN	MIN	MAX
SQL*Net message from client	984.010000	50.3%	95,161	0.010340	0.000000	0.310000
SQL*Net more data from client	418.820000	21.4%	3,345	0.125208	0.000000	0.270000
db file sequential read	279.340000	14.3%	45,084	0.006196	0.000000	0.050000
EXEC	136.880000	7.0%	67,888	0.002016	0.000000	1.320000
PARSE	74.490000	3.8%	10,098	0.007377	0.000000	0.090000
FETCH	37.320000	1.9%	57,217	0.000652	0.000000	0.130000
latch free	23.690000	1.2%	34,695	0.000683	0.000000	0.080000
log file sync	1.090000	0.1%	506	0.002154	0.000000	0.050000
SQL*Net more data to client	0.830000	0.0%	15,982	0.000052	0.000000	0.020000
log file switch completion	0.280000	0.0%	3	0.093333	0.080000	0.110000
enqueue	0.250000	0.0%	106	0.002358	0.000000	0.020000
SQL*Net message to client	0.240000	0.0%	95,161	0.000003	0.000000	0.010000
buffer busy waits	0.220000	0.0%	67	0.003284	0.000000	0.020000
db file scattered read	0.010000	0.0%	2	0.005000	0.000000	0.010000
SQL*Net break/reset to client	0.000000	0.0%	2	0.000000	0.000000	0.000000
TOTAL (15)	1,957.470000	100.0%	425,317	0.004602	0.000000	1.320000

- Quiz: What's the task's bottleneck?

CALL-NAME	DURATION	%	CALLS	MEAN	MIN	MAX
SQL*Net message from client	984.010000	50.3%	95,161	0.010340	0.000000	0.310000
SQL*Net more data from client	418.820000	21.4%	3,345	0.125208	0.000000	0.270000
db file sequential read	279.340000	14.3%	45,084	0.006196	0.000000	0.050000
EXEC	136.880000	7.0%	67,888	0.002016	0.000000	1.320000
PARSE	74.490000	3.8%	10,098	0.007377	0.000000	0.090000
FETCH	37.320000	1.9%	57,217	0.000652	0.000000	0.130000
latch free	23.690000	1.2%	34,695	0.000683	0.000000	0.080000
log file sync	1.090000	0.1%	506	0.002154	0.000000	0.050000
SQL*Net more data to client	0.830000	0.0%	15,982	0.000052	0.000000	0.020000
log file switch completion	0.280000	0.0%	3	0.093333	0.080000	0.110000
enqueue	0.250000	0.0%	106	0.002358	0.000000	0.020000
SQL*Net message to client	0.240000	0.0%	95,161	0.000003	0.000000	0.010000
buffer busy waits	0.220000	0.0%	67	0.003284	0.000000	0.020000
db file scattered read	0.010000	0.0%	2	0.005000	0.000000	0.010000
SQL*Net break/reset to client	0.000000	0.0%	2	0.000000	0.000000	0.000000
TOTAL (15)	1,957.470000	100.0%	425,317	0.004602	0.000000	1.320000

bottleneck误区

- bottleneck != busiest resource on the system while your task runs

- Bottleneck is defined in the context of a task execution.
- Different experiences on a given system have different bottleneck.

Amdahl's Law

- A task's response time can improve only in proportion to how much the task uses the thing you improve.

$$C(N) = \frac{N}{1+\alpha(N-1)}$$

	Response time		Potential improvement	Relative cost
1	1,748.229	70.8%	35.4%	1,000,000
2	338.470	13.7%	12.3%	1
3	152.654	6.2%		+∞
4	97.855	4.0%	4.0%	1
5	58.147	2.4%		+∞
6	48.274	2.0%	1.6%	1
7	23.481	1.0%		+∞
8	0.890	0.0%		+∞
	2,468.000	**100.0%**	**53.3%**	**1,000,003**

	Response time		Potential improvement	Relative cost
1	1,748.229	70.8%	35.4%	1,000,000
2	338.470	13.7%	12.3%	1
3	152.654	6.2%		+∞
4	97.855	4.0%	4.0%	1
5	58.147	2.4%		+∞
6	48.274	2.0%	1.6%	1
7	23.481	1.0%		+∞
8	0.890	0.0%		+∞
	2,468.000	**100.0%**	**53.3%**	**1,000,003**

skew

- a non-uniformity in a list
- what fouls your ability to predict results

- Quiz: How much time will you save if you eliminate half of the 10,013,394 "dbfilesequential read" calls?

CALL-NAME	DURATION	%	CALLS	MEAN	MIN	MAX
db file sequential read	59,081.406102	76.6%	10,013,394	0.005900	0.000010	15.853019
log buffer space	6,308.758563	8.2%	9,476	0.665762	0.000004	1.010092
free buffer waits	4,688.730190	6.1%	200,198	0.023420	0.000004	1.021281
EXEC	4,214.190000	5.5%	36,987	0.113937	0.000000	5.400000
log file switch completion	1,552.471890	2.0%	1,853	0.837815	0.000006	1.013093
22 others	1,303.198855	1.7%	70,400	0.000402	0.000000	8.964706
TOTAL (27)	77,148.755600	100.0%	10,332,308	0.007467	0.000000	15.853019

- Depends on which half?
- Eliminate the green ones (93% of calls), and you'll save 0.3% of time.
- Eliminate the red ones (just 5% of calls), and you'll save 98.5% of time.

	RANGE {min ≤ e < max}		DURATION	%	CALLS	MEAN	MIN	MAX
1.	0.000000	0.000001						
2.	0.000001	0.000010						
3.	0.000010	0.000100	199.445978	0.3%	9,346,059	0.000021	0.000010	0.000099
4.	0.000100	0.001000	21.420428	0.0%	108,351	0.000198	0.000100	0.000999
5.	0.001000	0.010000	612.513248	1.0%	106,319	0.005761	0.001000	0.009999
6.	0.010000	0.100000	11,193.505611	18.9%	314,869	0.035550	0.010000	0.099999
7.	0.100000	1.000000	26,057.804096	44.1%	130,471	0.199721	0.100002	0.999717
8.	1.000000	10.000000	20,804.497660	35.2%	7,308	2.846811	1.000184	9.900656
9.	10.000000	100.000000	192.219083	0.3%	17	11.307005	10.242772	15.853019
10.	100.000000	1,000.000000						
11.	1,000.000000	+∞						
		TOTAL (11)	59,081.406102	100.0%	10,013,394	0.005900	0.000010	15.853019

Minimizing Risk

- Efficiency
  - an inverse measure of waste
- waste
  - any work that could be eliminated without sacrificing useful benefits

- To measure efficiency, profile.

- The fastest way to do something is to not do it at all.

Load

- competition for a resource by concurrent task executions

- busier -> more waiting
  - queueing delay
  - coherency delay

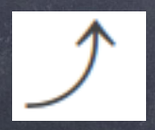
Queueing Delay

- time spent waiting in a queue for access to a shared resource

coherency delay

- time spent communicating and coordinating access to a shared resource

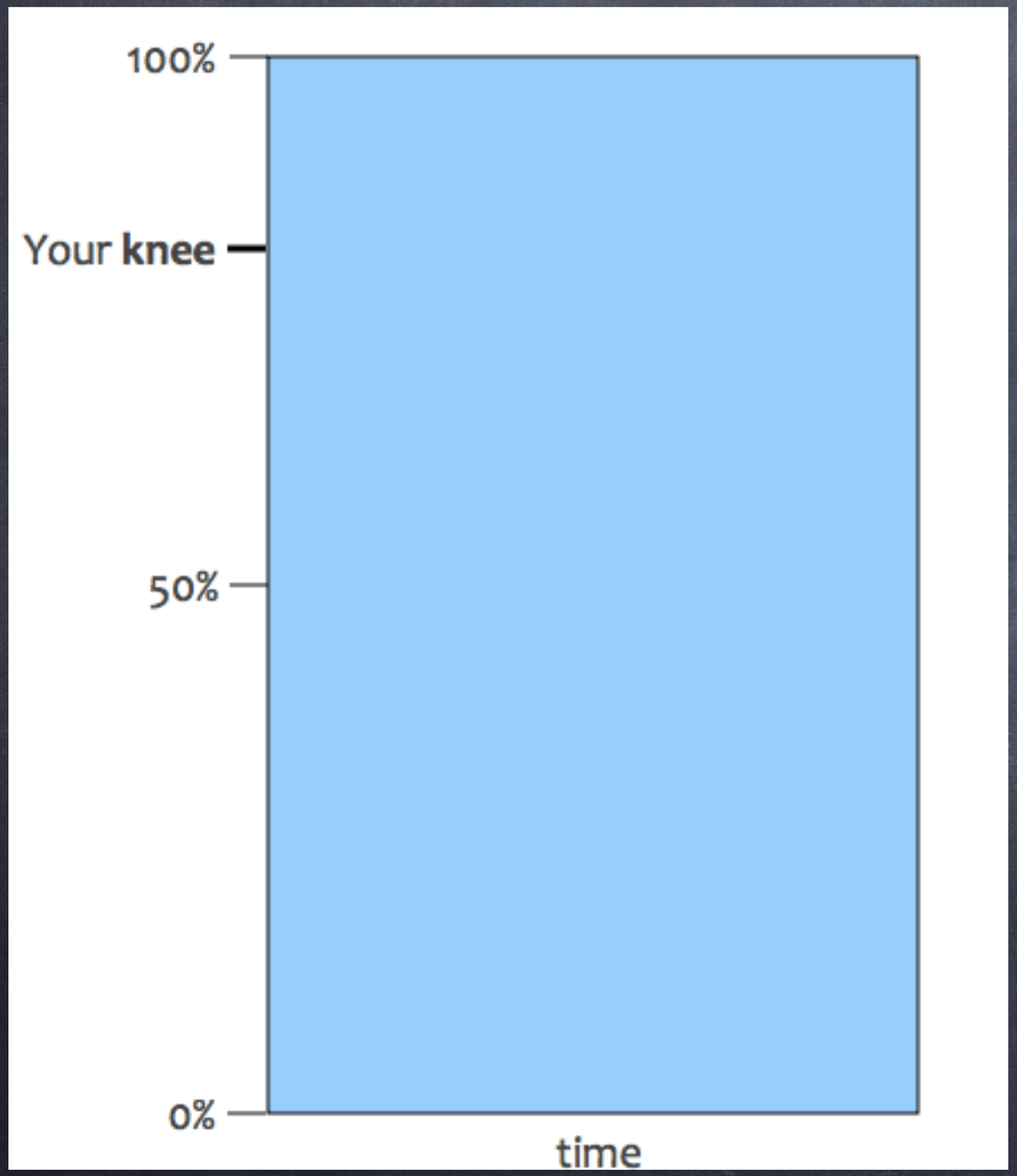
The Knee

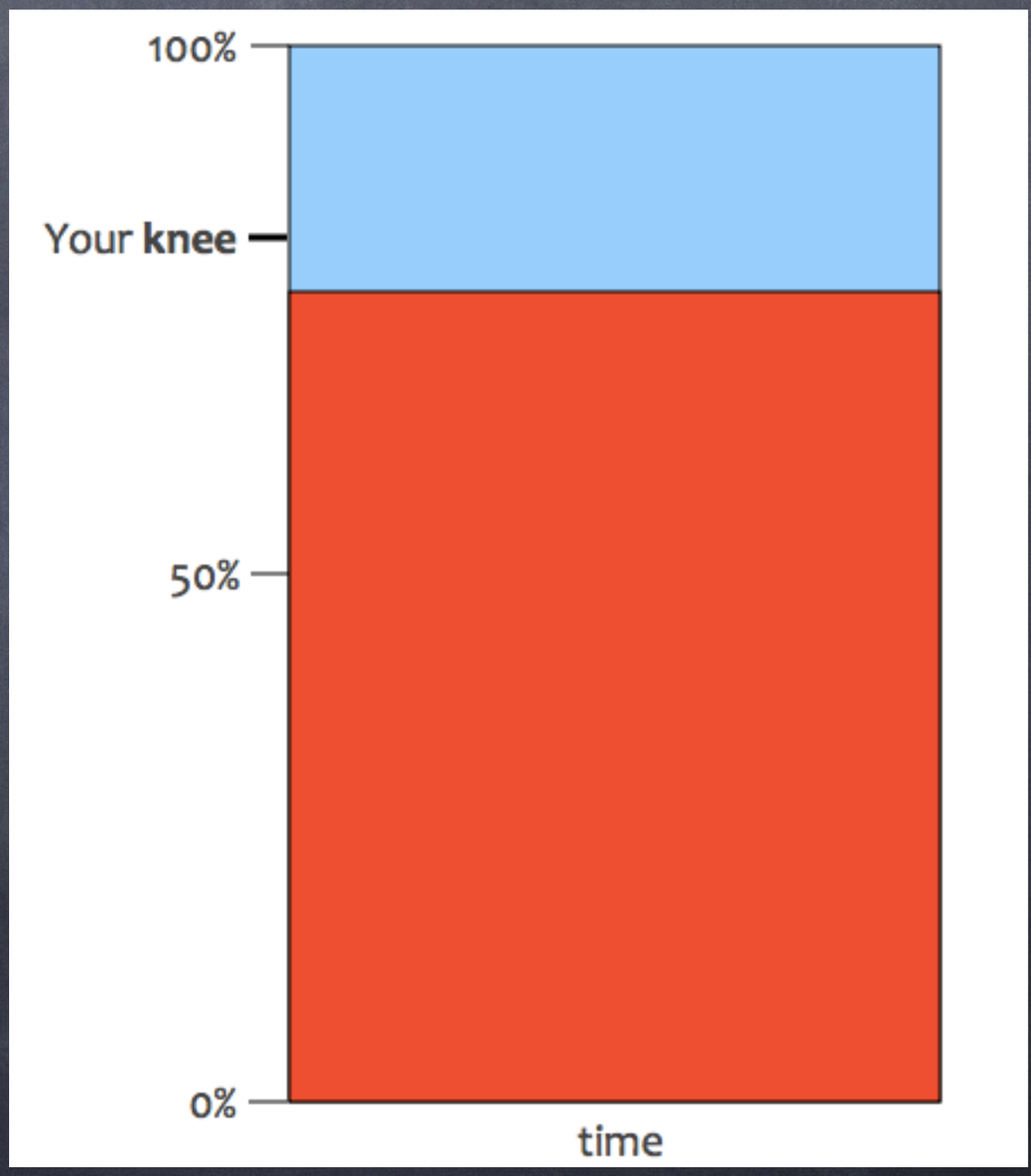
- Goals
  - response time 
  - throughput 

- the resource utilization value at which throughput and response time are in optimal balance

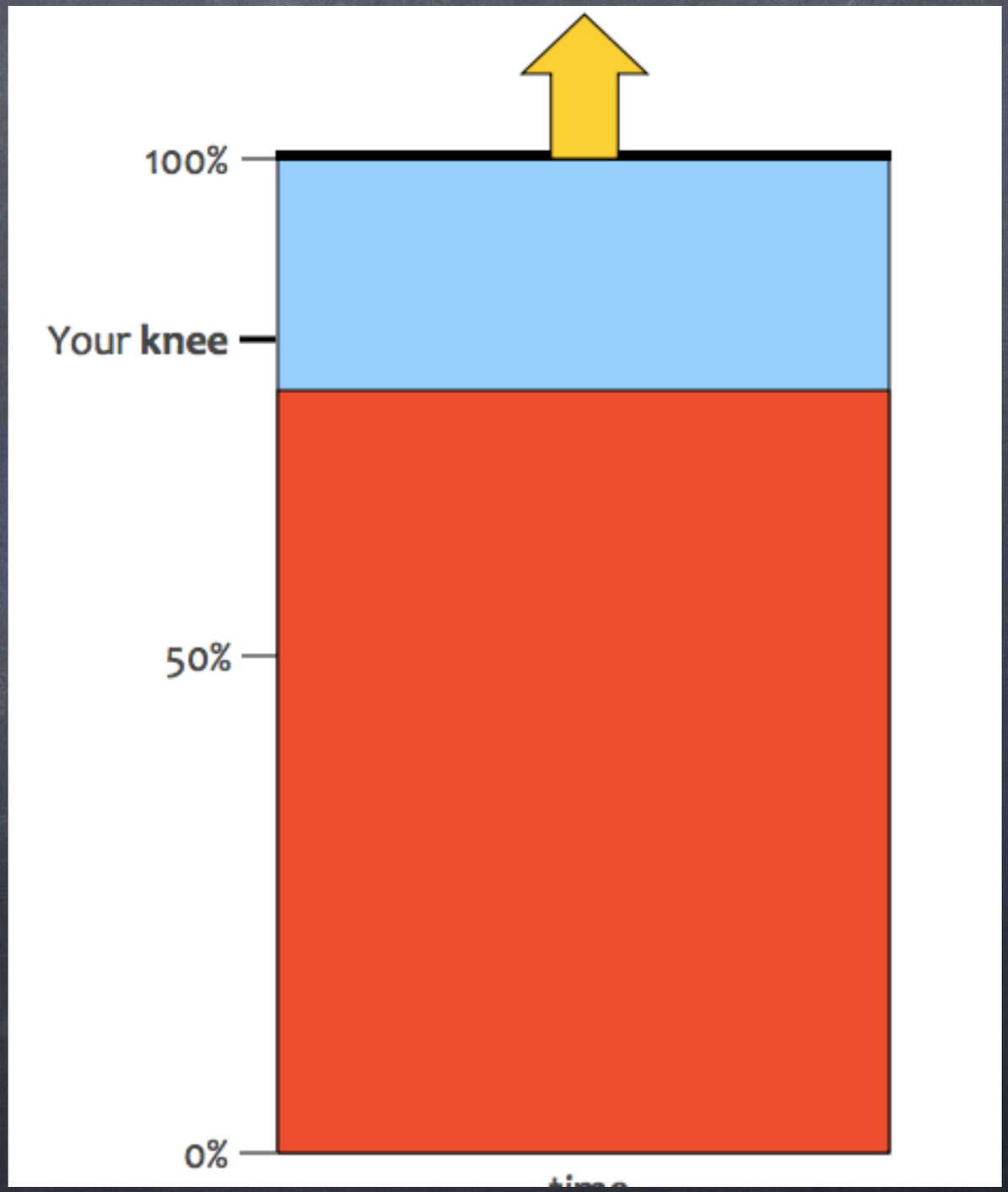
- Every resource has a **knee**.

Capacity Planning

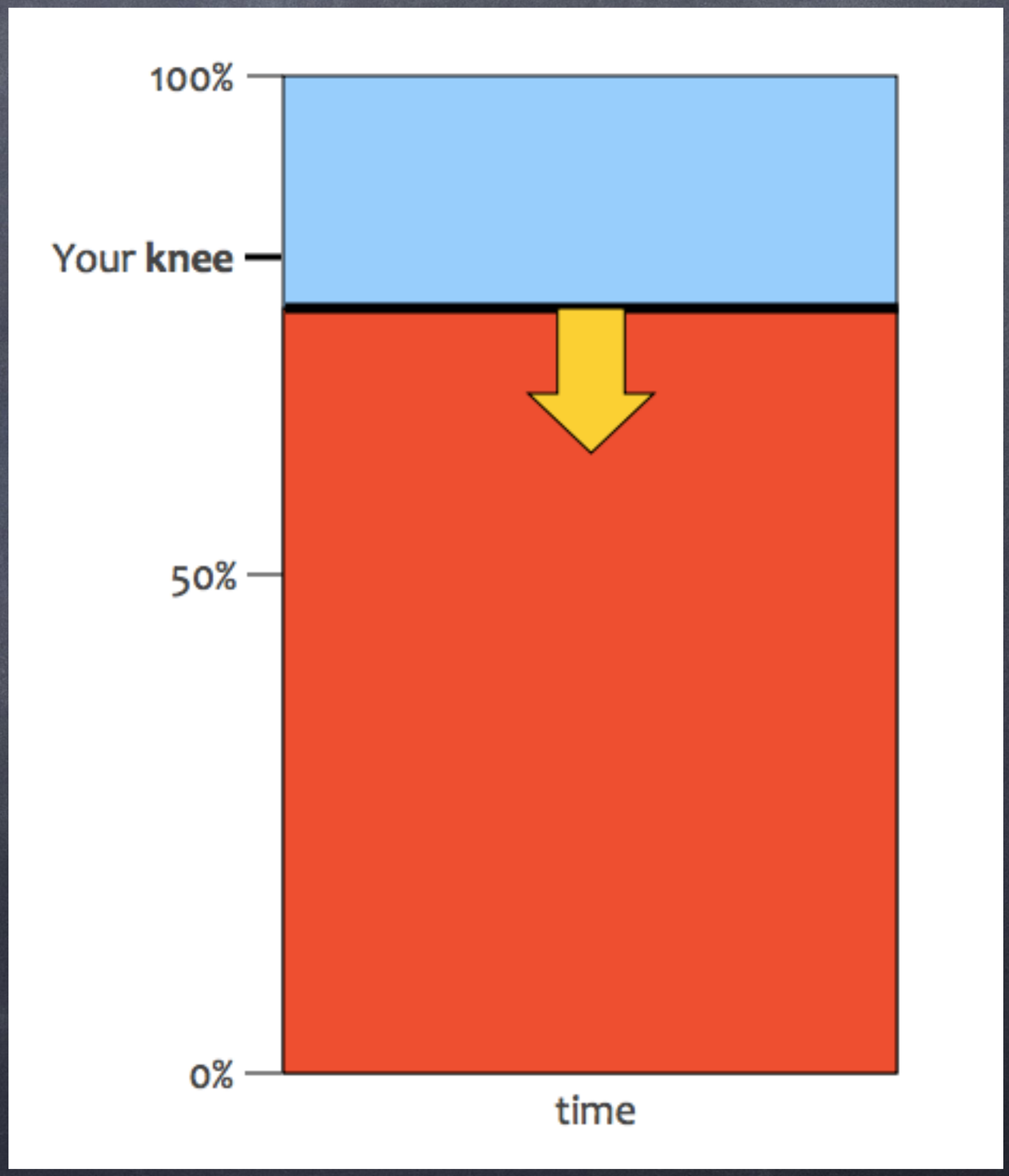




- capacity planning = How big must capacity be?



- load management = How small must load be?



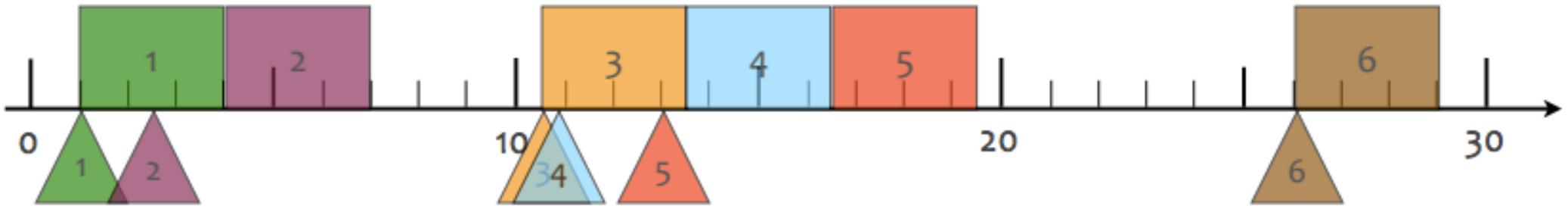
- To perform well, you must
- manage your load
- so that utilizations on resources with arrivals
- do not exceed their knees.

- When load exceed a knee, you need to
  - reschedule load,
  - or eliminate load,
  - or increase capacity.

random arrivals

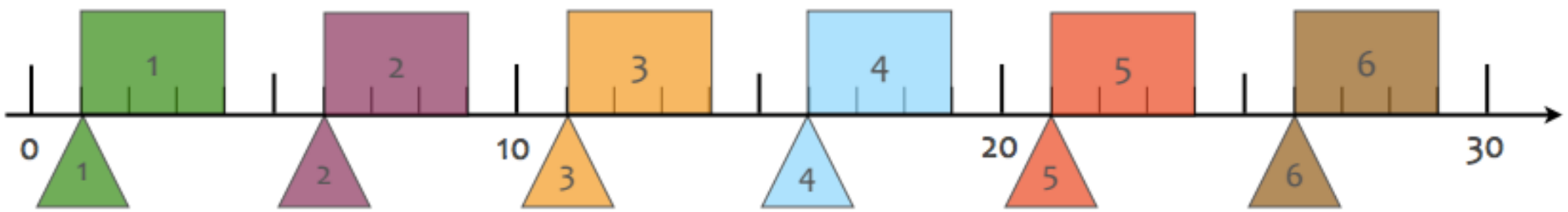
Random arrival process

A = 6 arrivals, T = 30 sec, λ = .2 arrivals/sec, S = 3 sec, **R = 4.267 sec**



Deterministic arrival process

A = 6 arrivals, T = 30 sec, λ = .2 arrivals/sec, S = 3 sec, **R = 3.000 sec**



- deterministic arrivals:
  - you can run up to 100% utilization
- random arrivals:
  - you must pay attention to your knees.

Performance Test

- Plan
- Measure
- tools

tools

- CPU消耗分析
- 内存消耗分析
- GC/heap消耗分析

- Load Test
- HTTP/TCP/UDP service
- http_load/JMeter

Thanks.