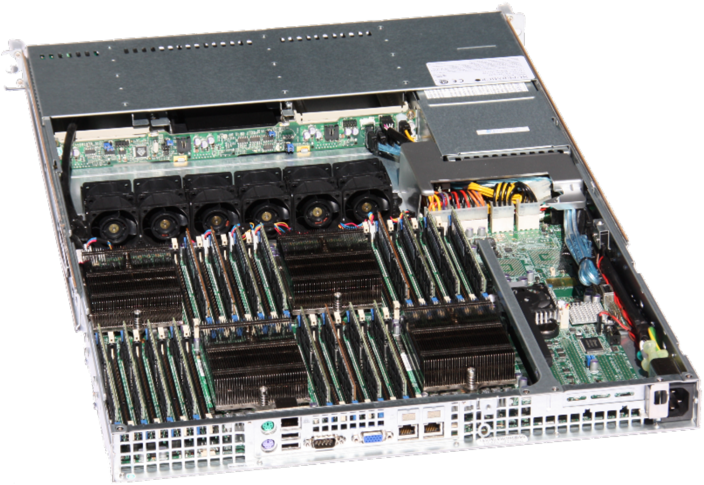
Analyzing Erlang with big data techniques

Ying Li
Concurix Corporation

March 21, 2013

Recap: The Manycore era is here now



AMD Opteron family 15h
64 cores: 16 per chip x 4 sockets
Streaming SIMD extensions (SSE4)
128 GB RAM—512GB max
8 NUMA Domains with Hypertransport
~$4.7K

Locks in traditional O-O languages limit scalability per Amdahl's Law

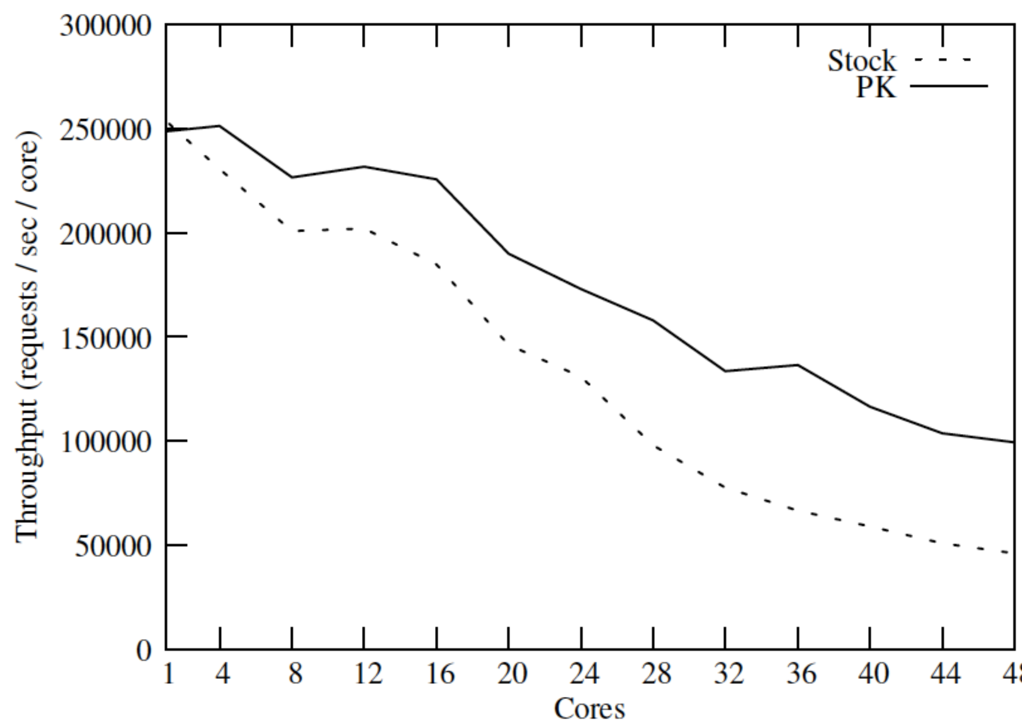
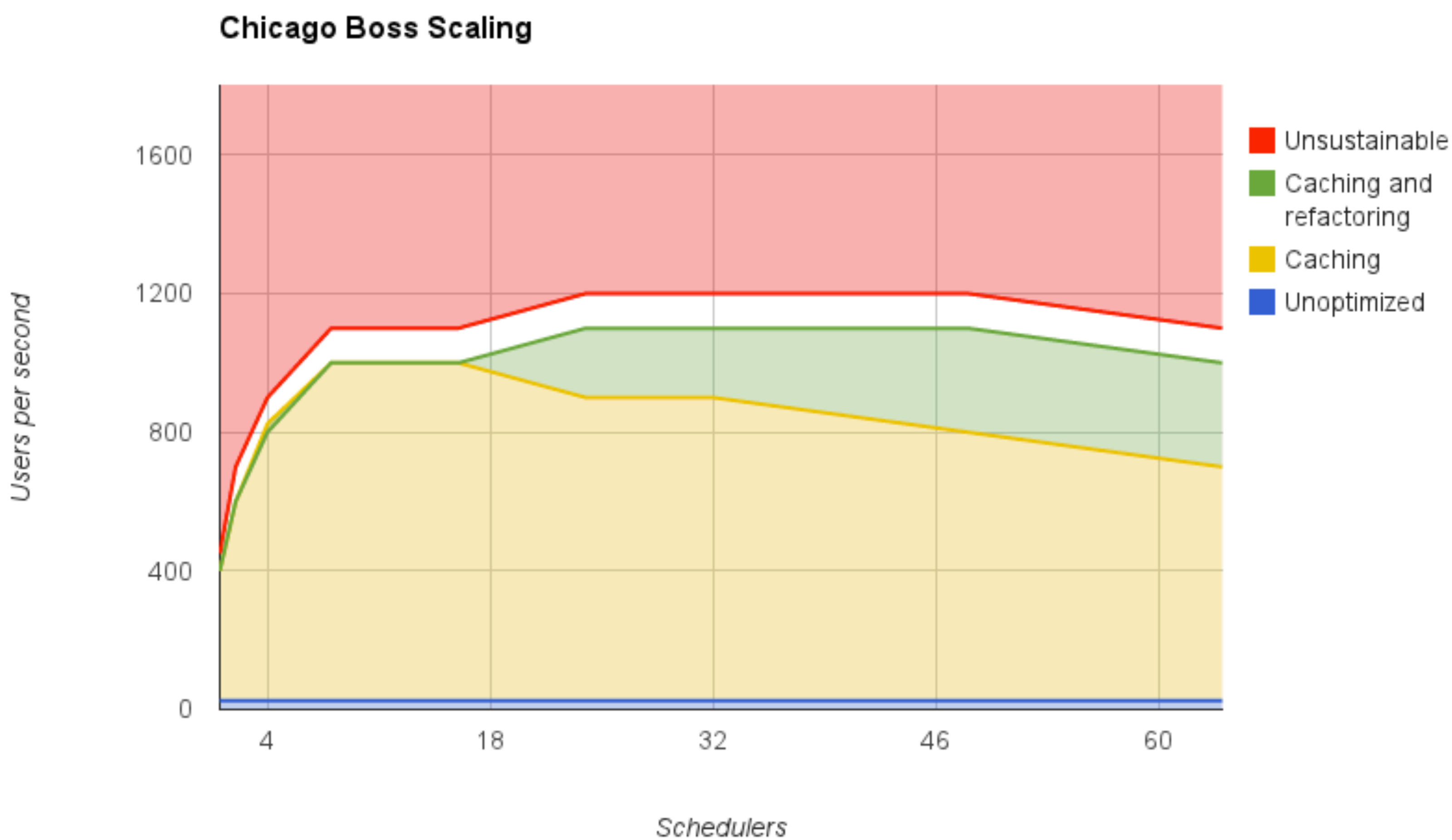
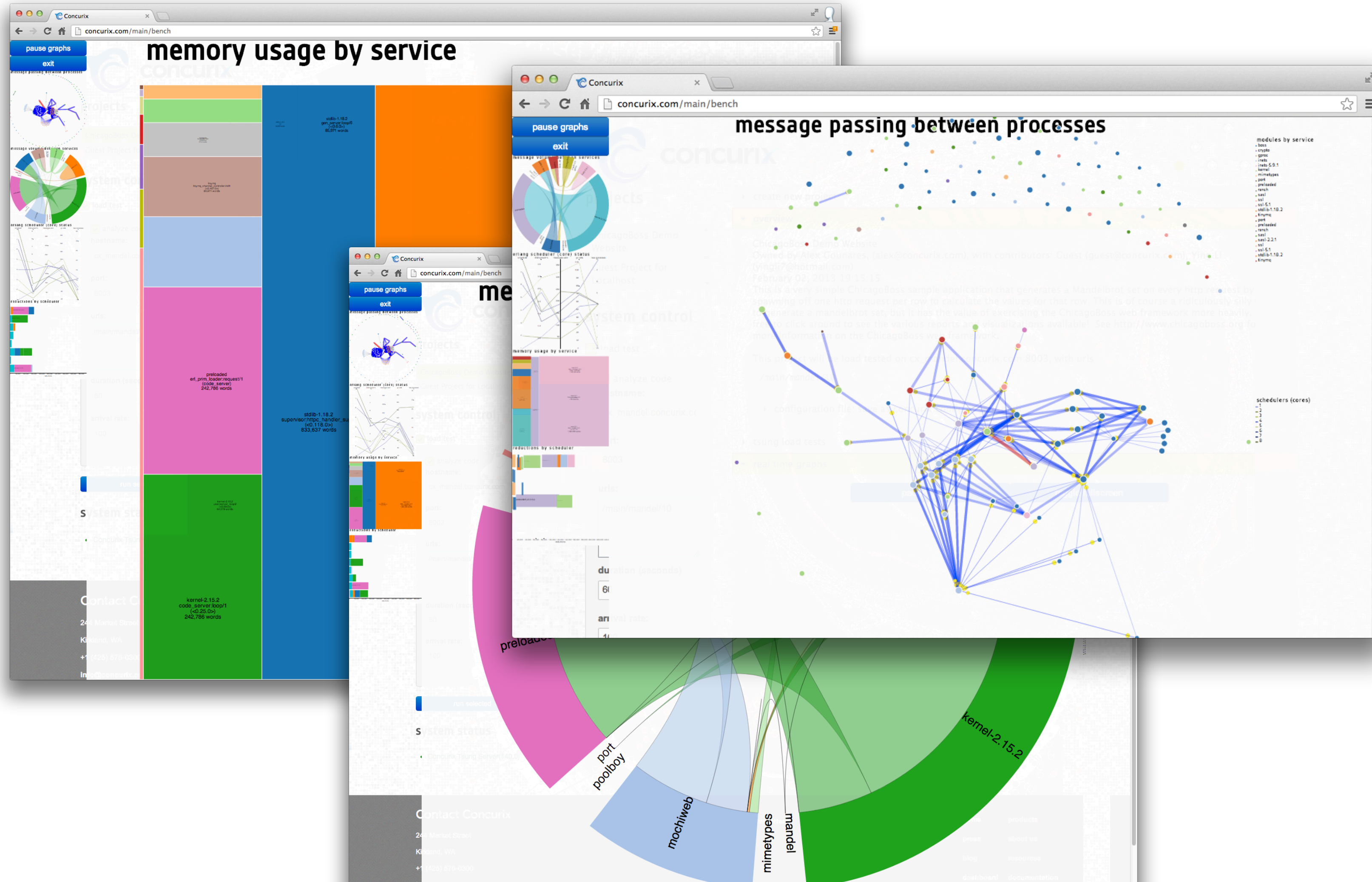


Figure 5: memcached throughput.

Recap: The Concurix Opportunity: Realizing Moore's Law for software



Recap: Summarize and animate application trace data



Data details and inspiration

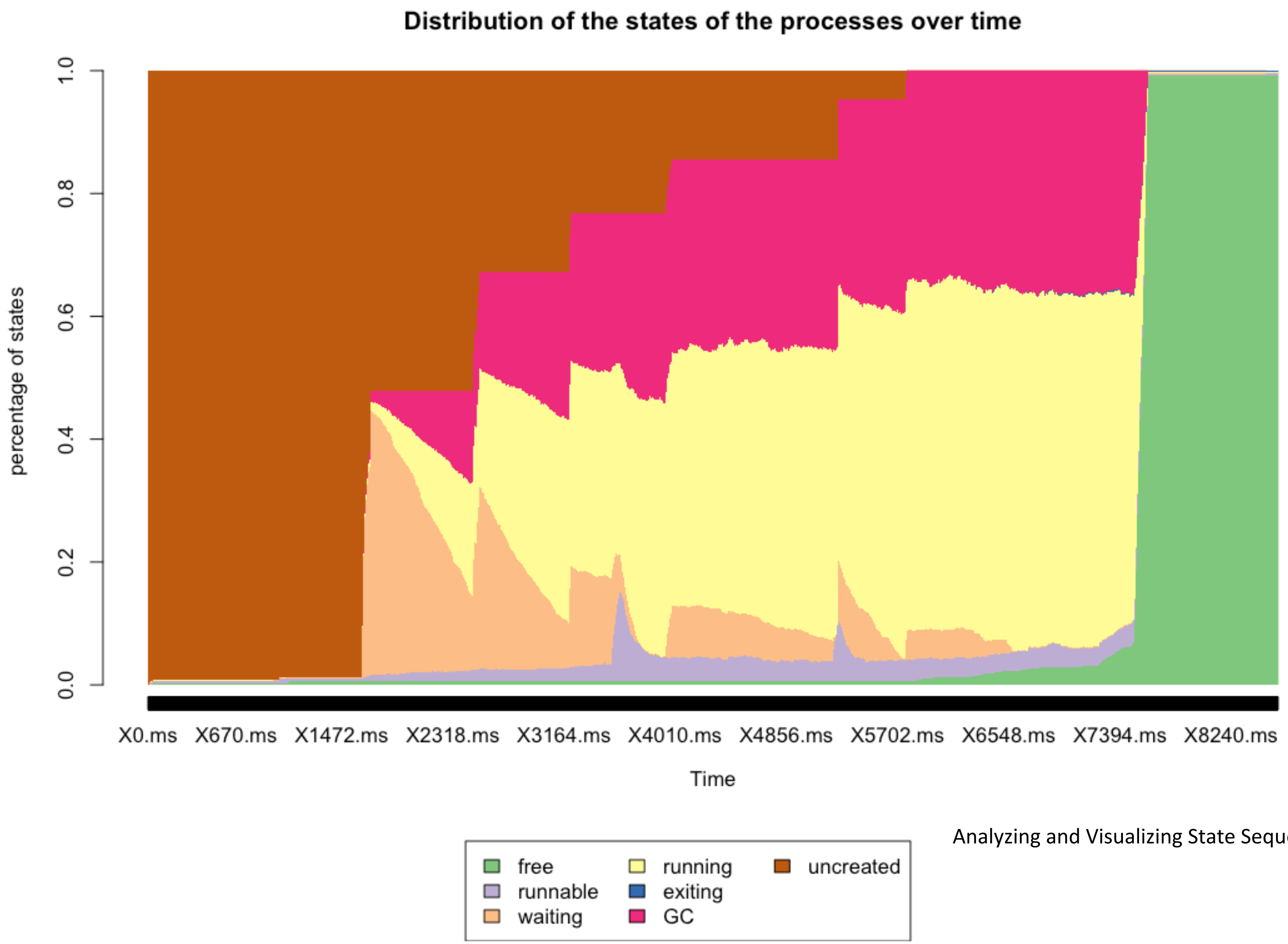
- on processes
  - ID, MFA, service, behavior
  - heap size
  - number of reductions
  - queue length
  - scheduler they are on
- on message passing
  - source process
  - target process
  - number of messages
  - number of words sent
- on schedulers
  - processes created
  - quanta count and quanta time
  - number of GC
  - true call count and tail call count
  - return count
  - processes freed

- Data from many API's
- New instrumentation in VM to give us more data
- Brought all together in an organized manner that's easy to use

- 2-second snapshots stored in AWS S3 available for batch analytics and deeper analysis
- time dimension explodes the power of data

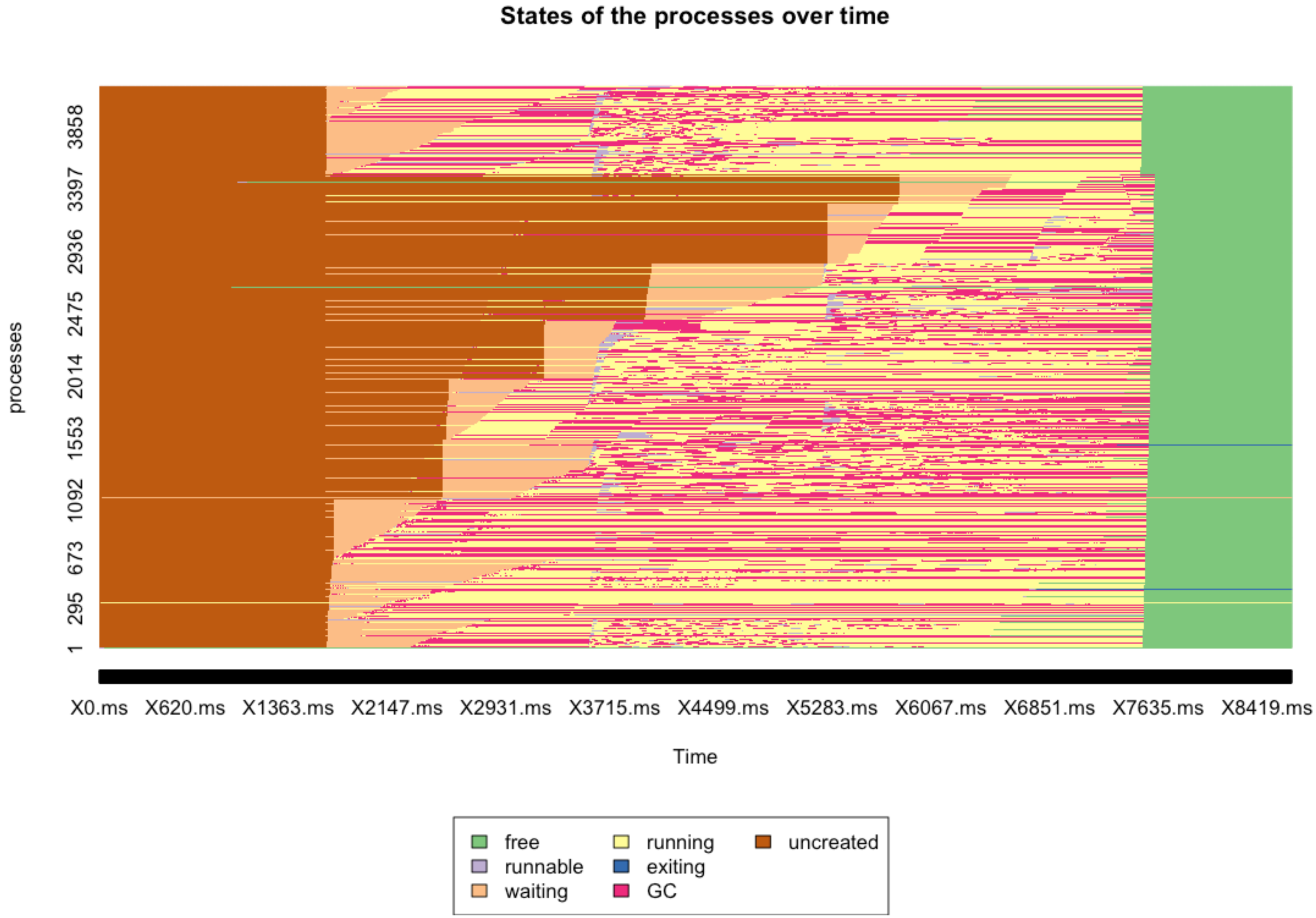
- change of scenery for a data miner from online advertising world
  - objects of study
    - Processes, schedulers, messages,
  - feedback loop
  - speed of instrumentation and data generation

Process state distribution over time

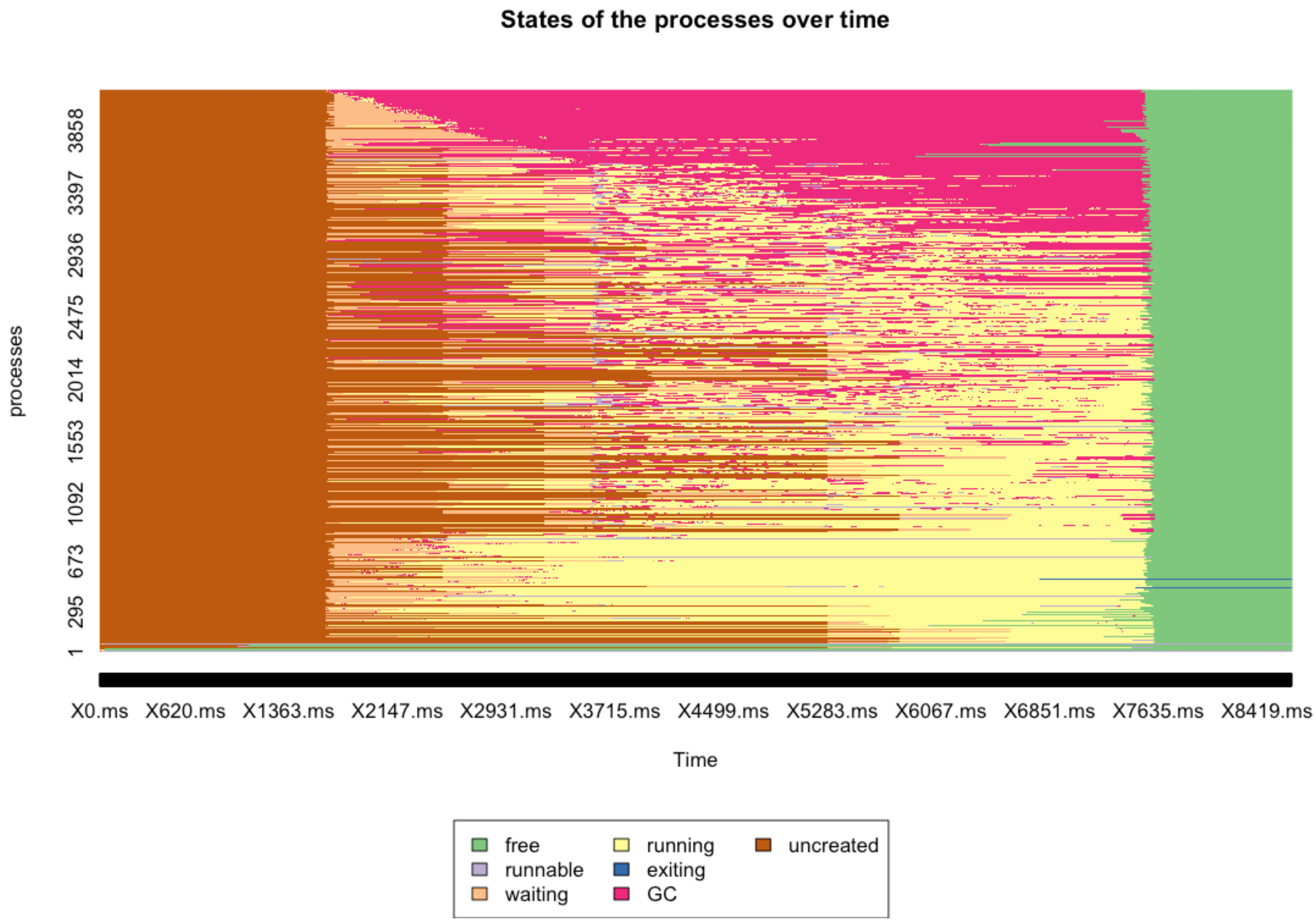


Analyzing and Visualizing State Sequences in R with TraMineR

Process sequences over time



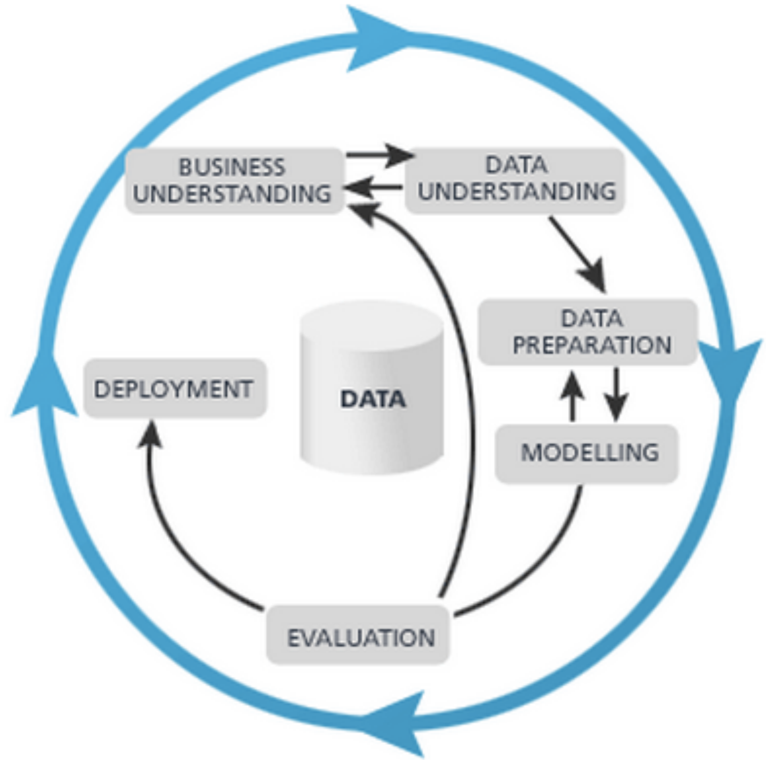
Looking at the data slightly differently



Data Mining Opportunities

- Data Mining - the non-trivial extraction of novel, implicit, and actionable knowledge from large datasets
  - Extremely large datasets
  - Discovery of the non-obvious
  - Useful knowledge that can improve processes
  - Can not be done manually

- Data mining helps us identify problem areas in ways we previously could not do
  - Lots of interacting parts, doing traces of all sorts, gaining understanding by analyzing the data
  - Instrumenting the interpreter and compiler



Example

programmer stepping through code in the debugger – this does not scale to large number of processes or large software

Measuring Similarity

- If we line up the data in an ordered vector, treat the vector as a point in N dimensional space, then similarity between data sets can be measured by distance between the points
  - One implementation is the cosine similarity

$$\theta = arccos\left(\frac{a \cdot b}{\sqrt{\sum_{k=1}^{n} a^2{}_k}\sqrt{\sum_{k=1}^{n} b^2{}_k}} \right)$$

- Application example
  - Identify transitions
    - Data = the number of messages sent between any pair of sender-receiver, during a time window
    - Vector = line up the data along all possible pairs, , like "mochiweb-to-poolboy", in fixed order
    - Data for each time window is a vector of length of NxN (N = number of processes)
    - Big change in similarity between the data vectors means big shift in message passing activity
  - Benchmark repeatability

```
1.000 0.999 0.923 0.999 benchrun-399
----- 1.000 0.917 0.999 benchrun-403
----- ----- 1.000 0.910 benchrun-404
----- ----- ----- 1.000 benchrun-406
```

Clustering

- Grouping data points by similarity on some metric
- Algorithm: repeat until converge
  - Assignment to clusters:

$$Cluster^{(t)}_i = \{ p : \|p - center^{(t)}_i\| \leq \|p - center^{(t)}_j\| \; \forall j \leq k \}$$

  - Update cluster centers

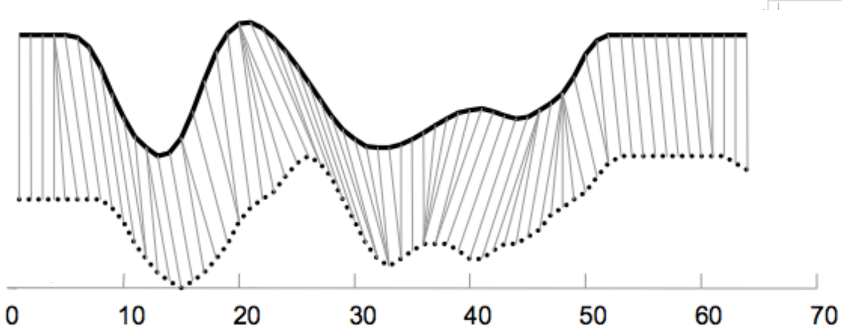
$$center^{(t+1)}_i = \frac{1}{|Cluster^{(t)}_i|} \sum_{p \in Cluster^{(t)}_i} p$$

- Applications
  - Often used for discovery tasks when there are little prior knowledge about data

Example: Bucket the many processes to a few types according to their activities over time

Time series analysis

- Use “dynamic time warping” distance to measure how well two time series match



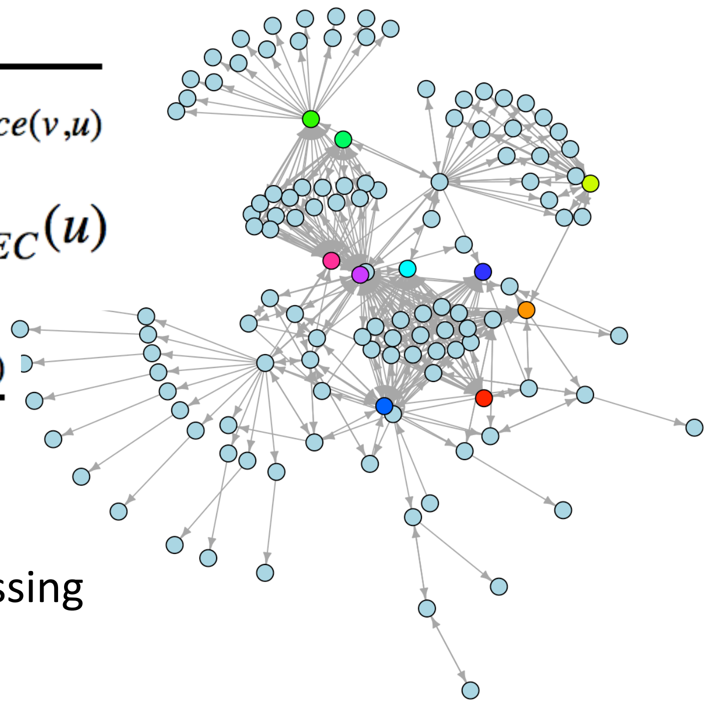
Diff	DTW
$a == b$	$\lvert a - b \rvert$
INSERT	Shift time out
DELETE	Shift time in

- Example:
  - Count some activity per second for each process.
  - Bucket processes that align well and focus the remaining outliers.

- Application: Time series clustering
  - Compute DTW distance
  - Apply regular or any clustering algorithm

Network analysis: centrality

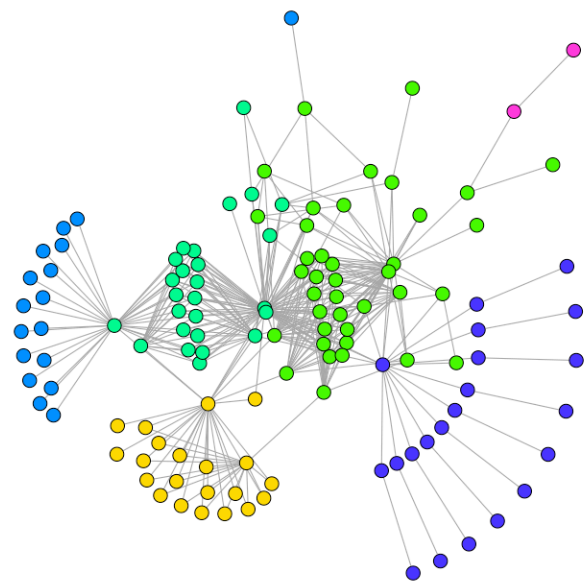
- Centrality: relative importance of a vertex in the network
- Computation
  - Degree centrality $C_D(v) = degree(v)$
  - Closeness centrality $C_C(v) = \frac{n-1}{\sum_{u \in V} shortest\text{-}distance(v,u)}$
  - Eigenvector centrality $C_{EC}(v) = \lambda \sum_{\{u,v\} \in E} C_{EC}(u)$
  - Betweenness centrality $C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$



- Application
  - Detecting most important process in message passing network
  - Potential for identifying bottlenecks?

Network analysis: community detection

- Communities: groups of vertices in the network that are “similar” to each other
- Algorithms
  - Minimum cut: partition the graph such that the number of edges across groups are minimal
  - Clustering of vertices
  - Vertex betweenness: the number of shortest path between pairs of vertices that run through it
    - Calculate betweenness for all
    - Remove the edge with highest betweenness
    - Recalculate betweenness for remaining vertices
    - Repeat until no edge remains



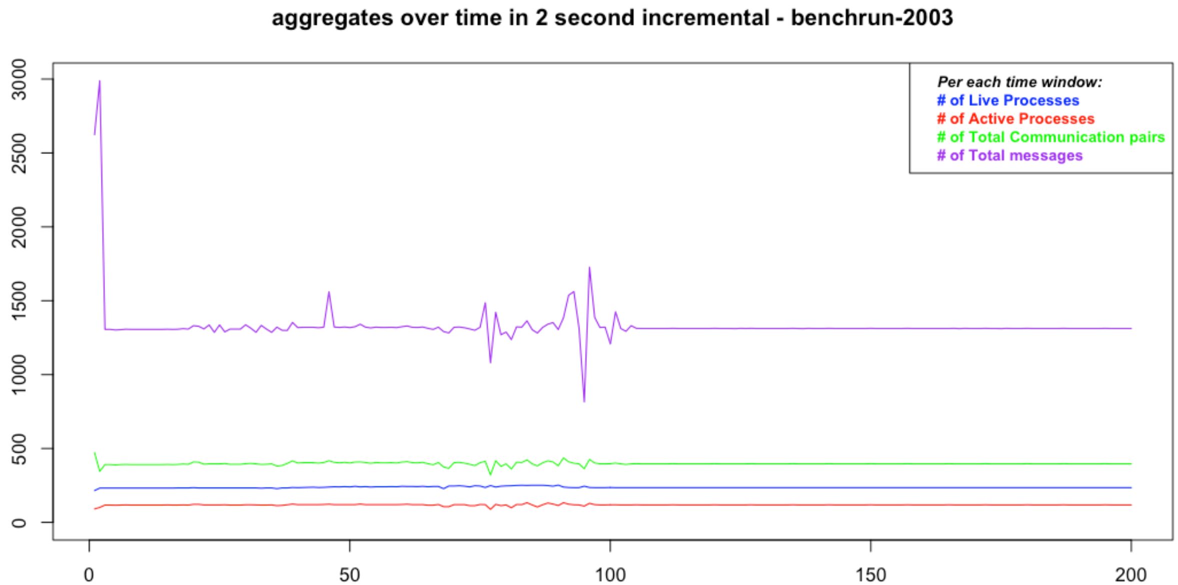
- Application
  - Detecting communities in message passing network
  - Potential for placing processes in the same community on the same core?

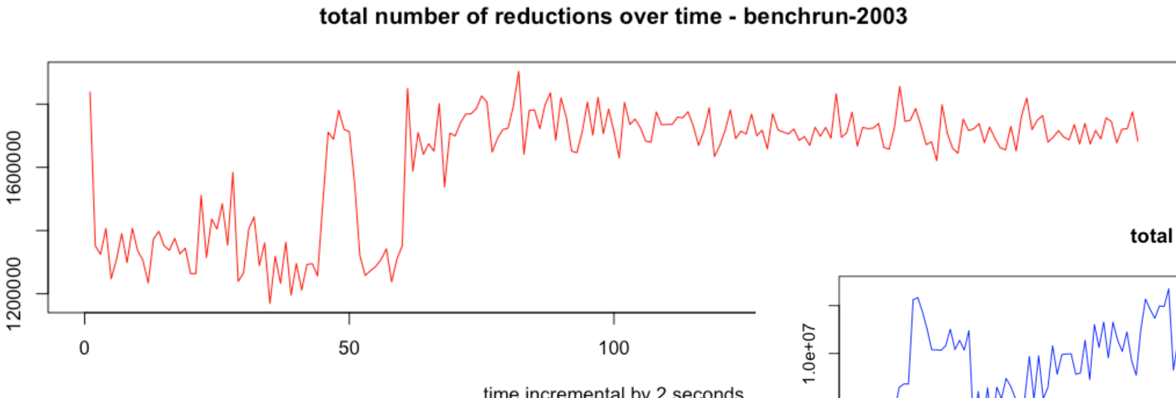
Applying big data techniques to applications

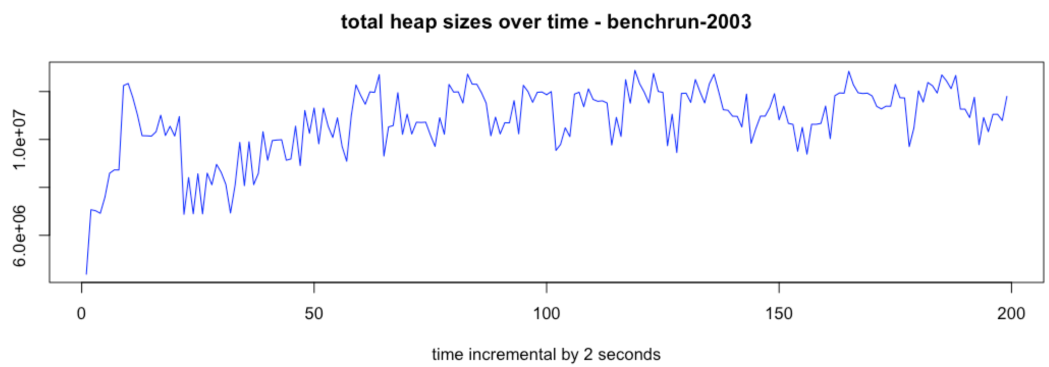
- early customers' data and data from our own applications and sample codes
- analyses influenced the design of our product but some specific analyses are not presented in the Concurix website yet
- all data are generated from the Concurix visualization AMI and Concurix Runtime
- all data captured and stored in AWS S3, same as any one using Concurix AMI to study their applications
- snapshot data captured at every 2 seconds
- One example: one production run has 17,939 snapshots

Applications: Concurix website under cyclic load

- www.concurix.com
- using ChicagoBoss
- moderate load of requests simulated by Mandelbrot patterns

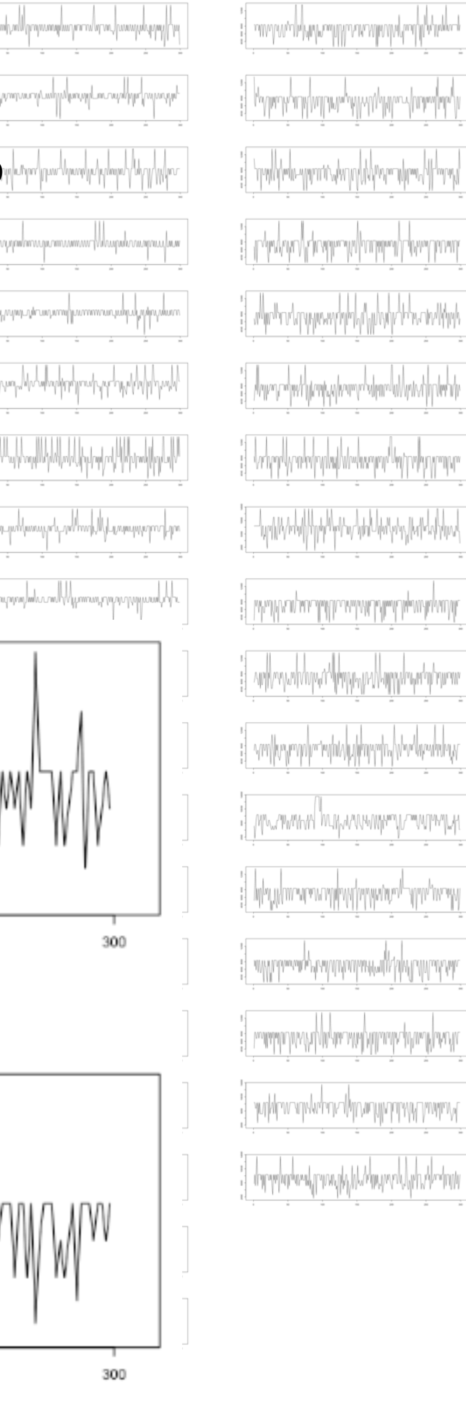


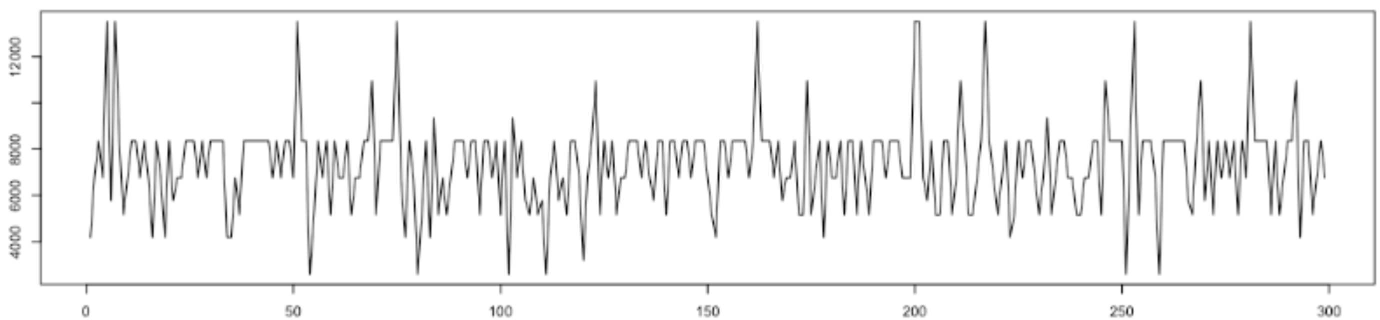


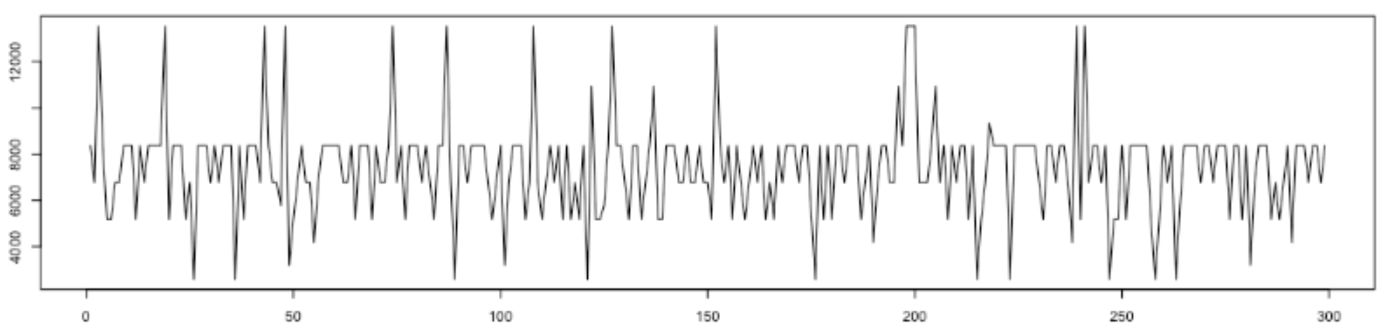


Study the heap sizes

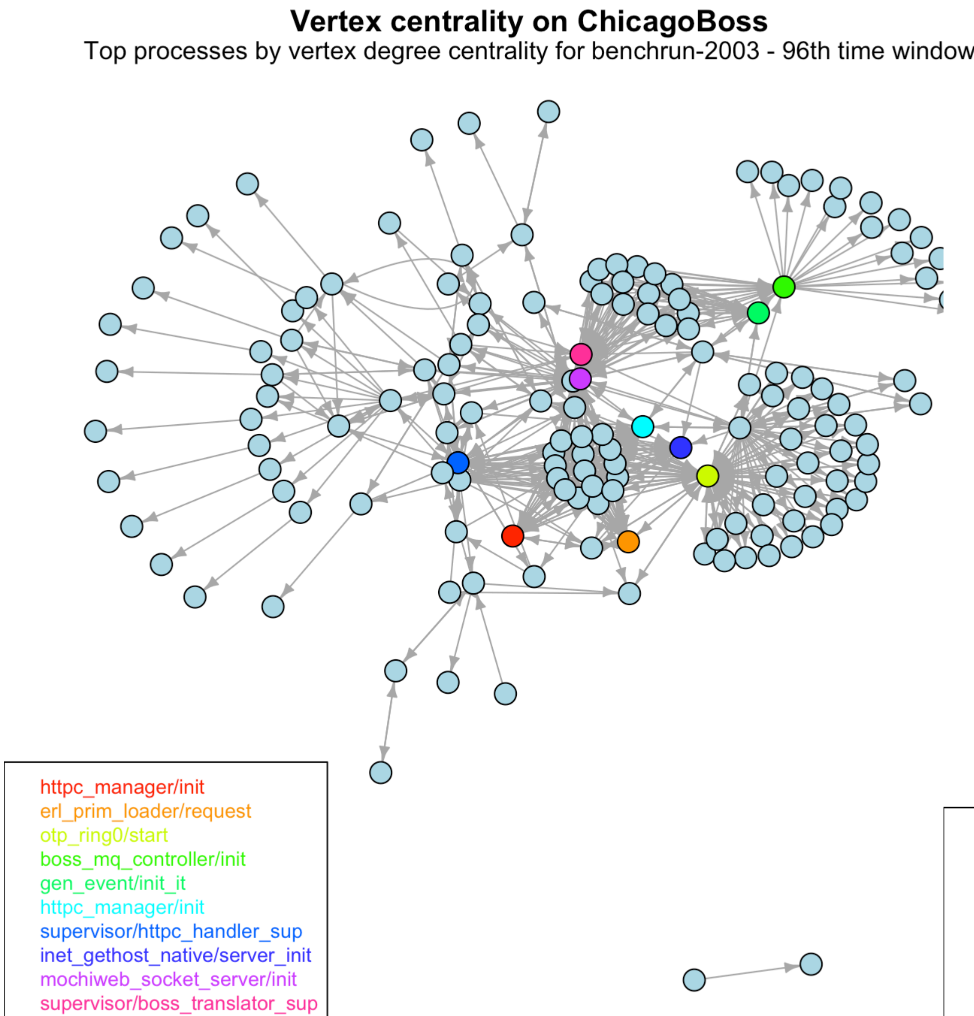
- Clustering the heap sizes over time as time series
  - > 9000 processes in the run, time series over 300 time snapshot windows
  - 5 clusters, map to MFA/service structure well, plus outliers
  - Similar time varying pattern within clusters

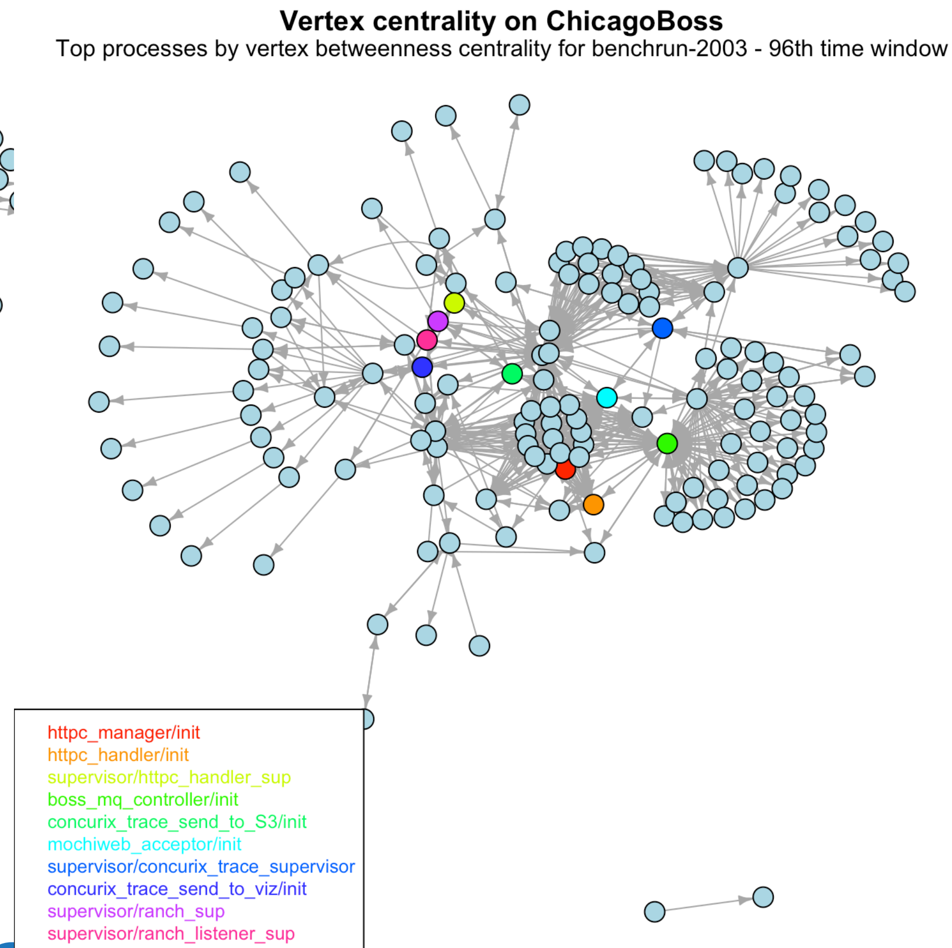






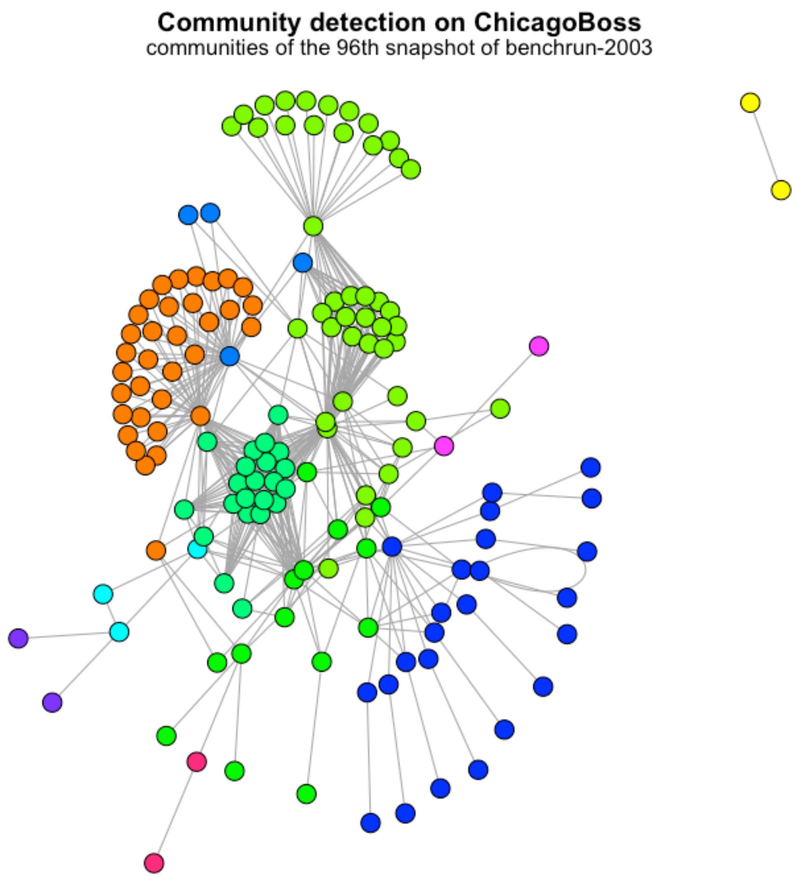
ChicagoBoss: network centrality

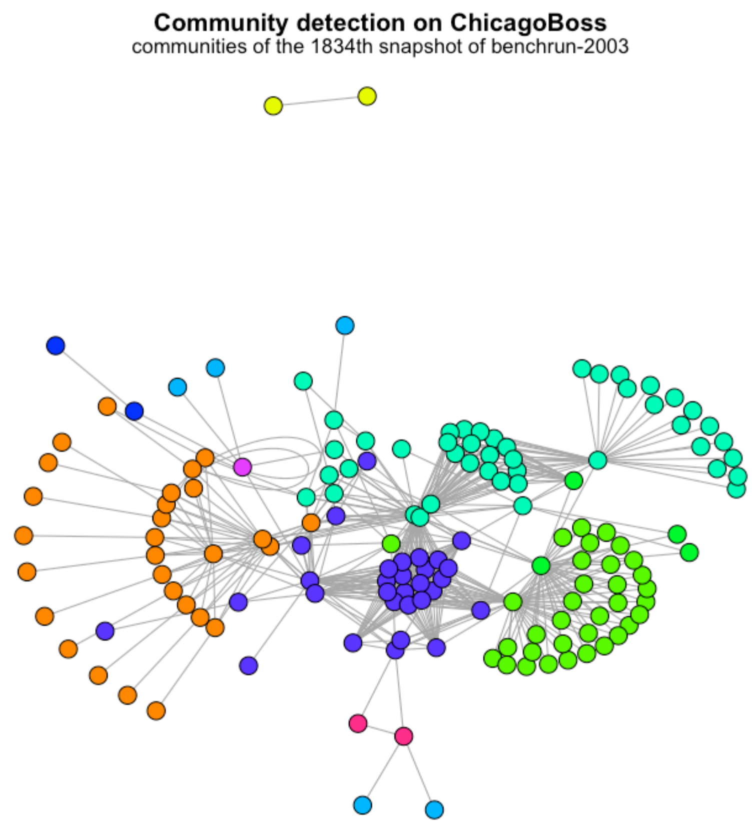




Study the structural changes

- At time window 96 (where the spike was) compared with time window 1834
  - More message volume but no structural differences observed

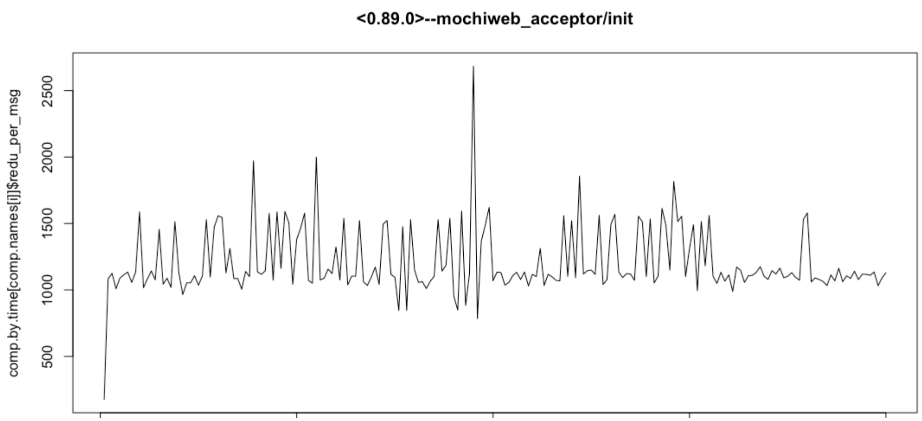


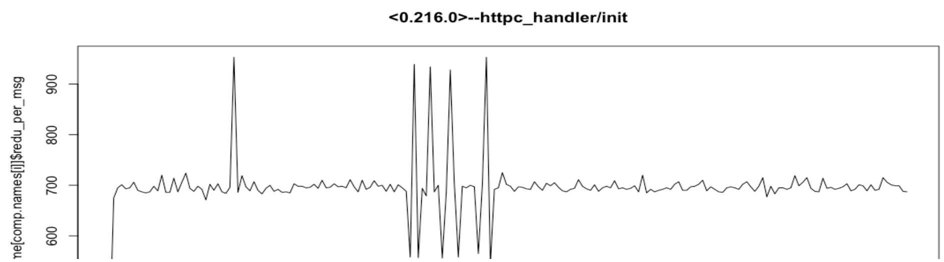


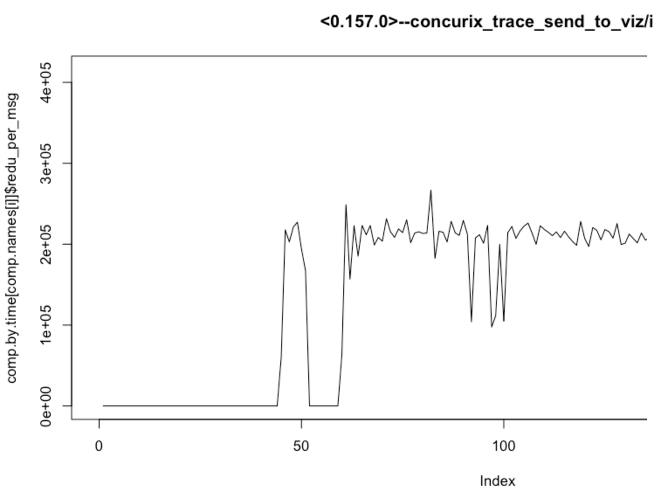
Detecting bottleneck based on trace data

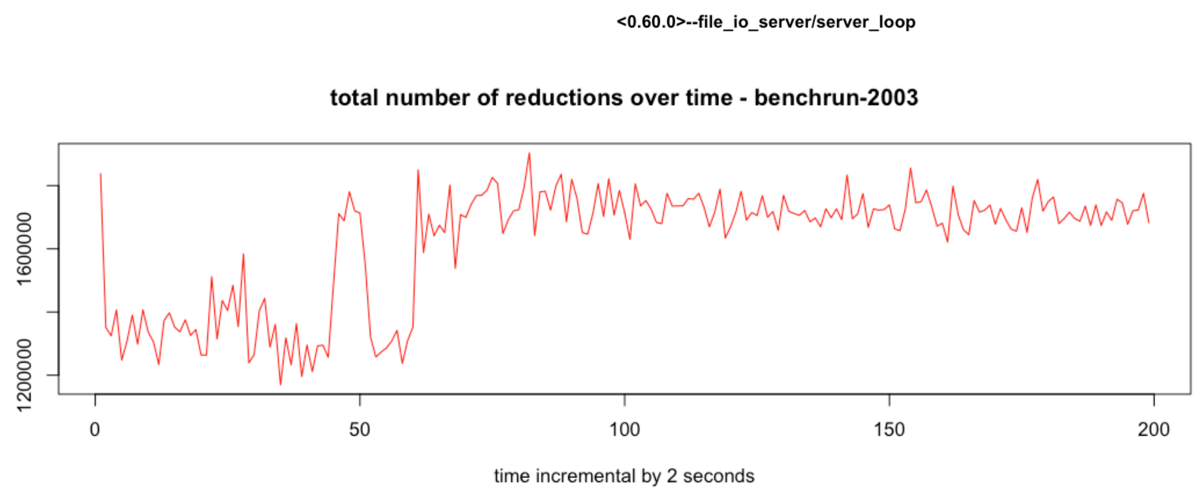
- Processes with highest number of reductions per incoming messages

[1] "mochiweb_acceptor/init"
[2] "timer/init"
[3] "concurix_trace_send_to_viz/init”
[4] "httpc_handler/init"
[5] "concurix_trace_send_to_S3/init”
[6] "user/server_loop"
[7] "file_io_server/server_loop"









Q&A