Uncertainty in Aggregate Metrics from Sampled Distributed Traces

2012 Workshop on Managing Systems Automatically and Dynamically

Nate Coehlo, **Arif Merchant**, Murray Stokely

Overview

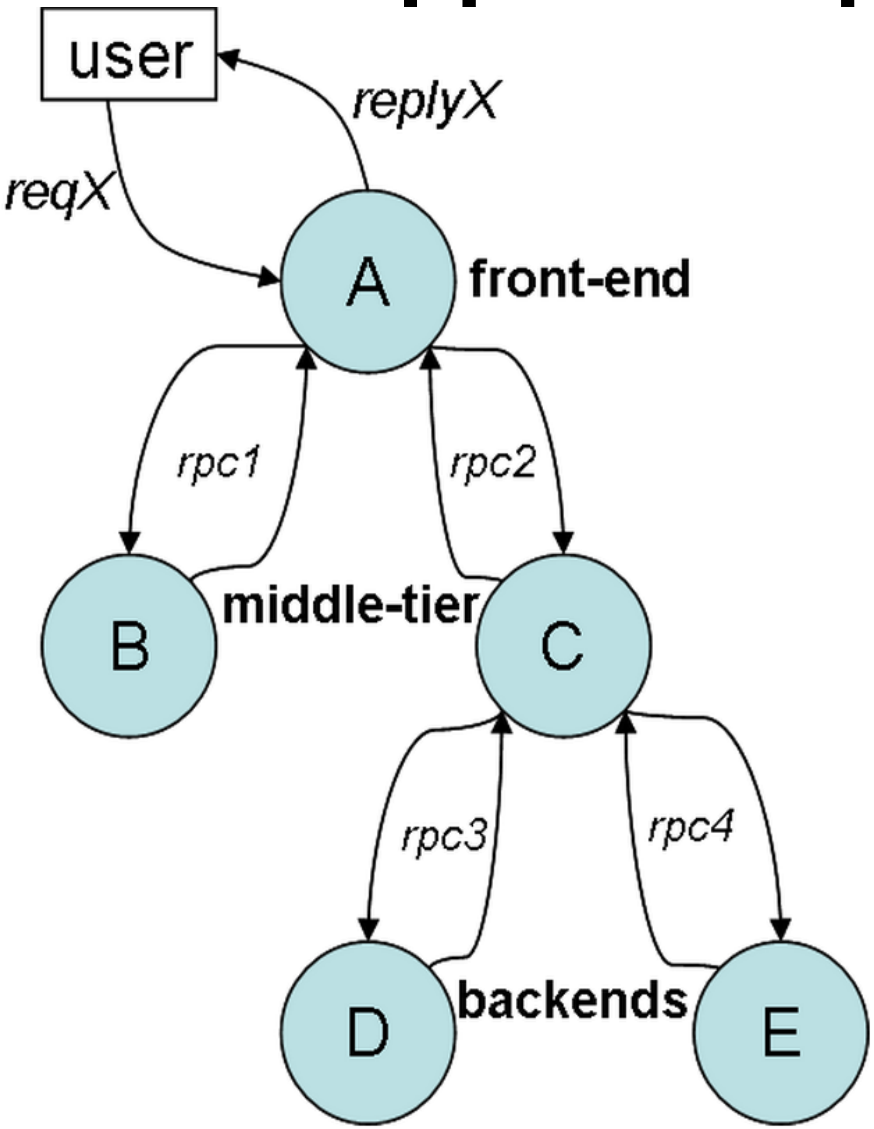
- Sometimes we lack a system measurement:
  - High measurement data volume.
  - Lack of perfect foresight / difficult implementation.

- Dapper: 'always-on' system for sampled distributed tracing.

- Can estimate metrics by aggregating Dapper samples.

- How to estimate the uncertainty in the aggregates?

Dapper Sampling 1: Overview



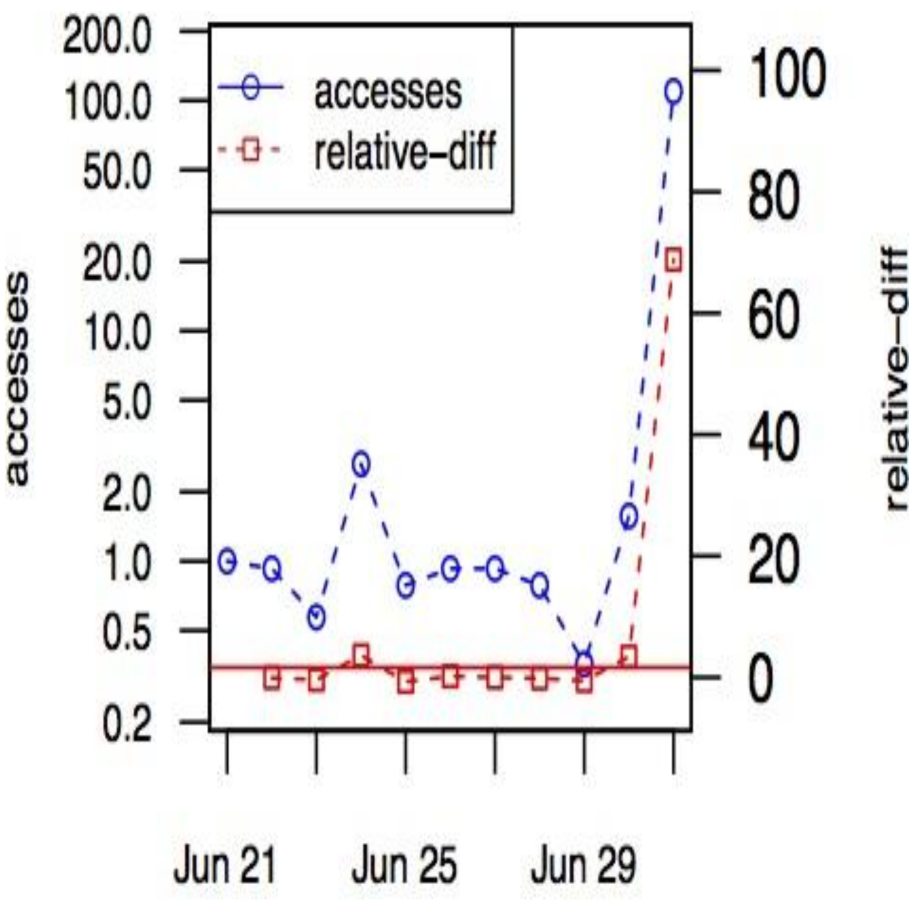
- **Simple Case:** Only complete traces returned, and all four RPCs have the same sampling probability.
- **Complication 1:** Developers may want more detailed information on middle-tier C, so they can configure this to make rpc3 and rpc4 get sampled with higher probability. This is **s**, the server sampling probability.
- **Complication 2:** Backend E might be under pressure so that collection needs to be further downsampled. This is **d**, the downsampling probability.
- For every RPC that gets returned, we also know the the sampling probability **p=s*d**.
- Doing weighted sums by **1/p** will give unbiased estimates.

* Figure from Sigelman et al, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure"

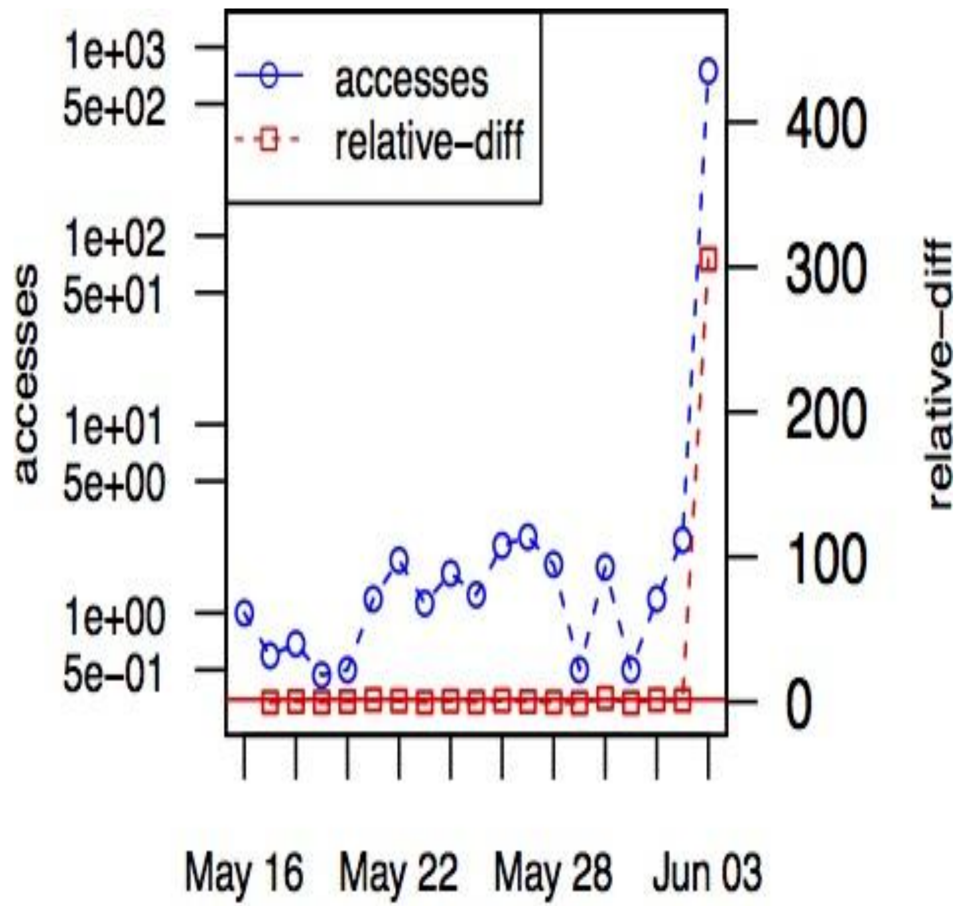
Example: Changes in Disk Accesses to certain data partitions

Both data partitions saw a large one day increase in the estimated number of disk seeks. When should we flag the difference?

Data Partition A: ~65x increase one day



Data Partition B: ~300x increase one day



Hypothesis Testing Approach

$$E_t = Estimated \# \, of \, disk \, accesses \, on \, day \, t$$

$$D_t = True \# \, of \, disk \, accesses \, on \, day \, t$$

Some natural variation exists, so our null hypothesis is:

$$H_0 : \; D_{t+1} < 1.1 \, D_t$$

We will reject the null hypothesis for large values of:

$$T = E_{t+1} - 1.1 \, E_t$$

A z-score is given if we divide T by its standard error.

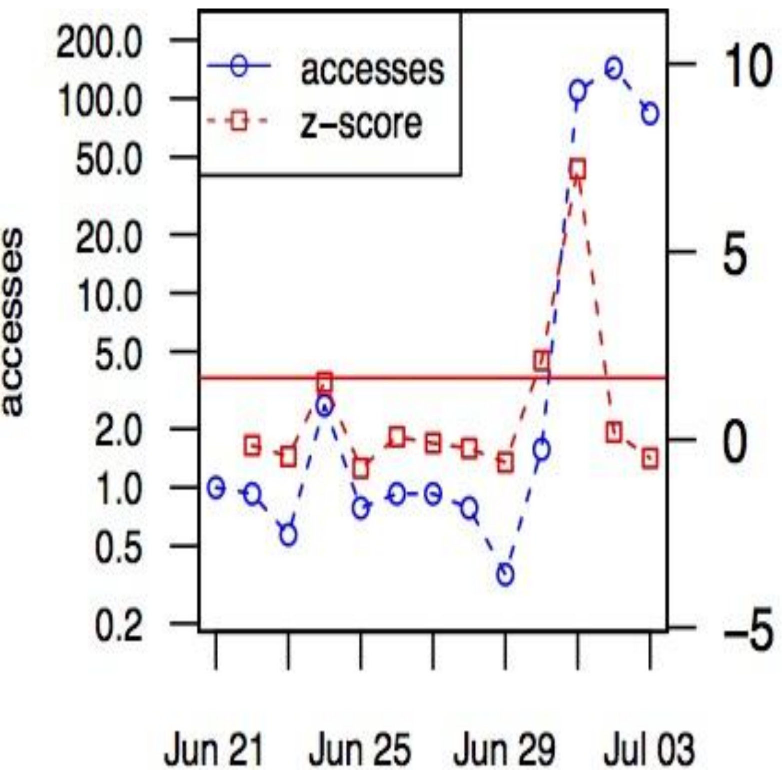
Based on the normal approximation, rejecting this one-sided null when z-score > 1.64 ensures a false positive rate of less than 5%.

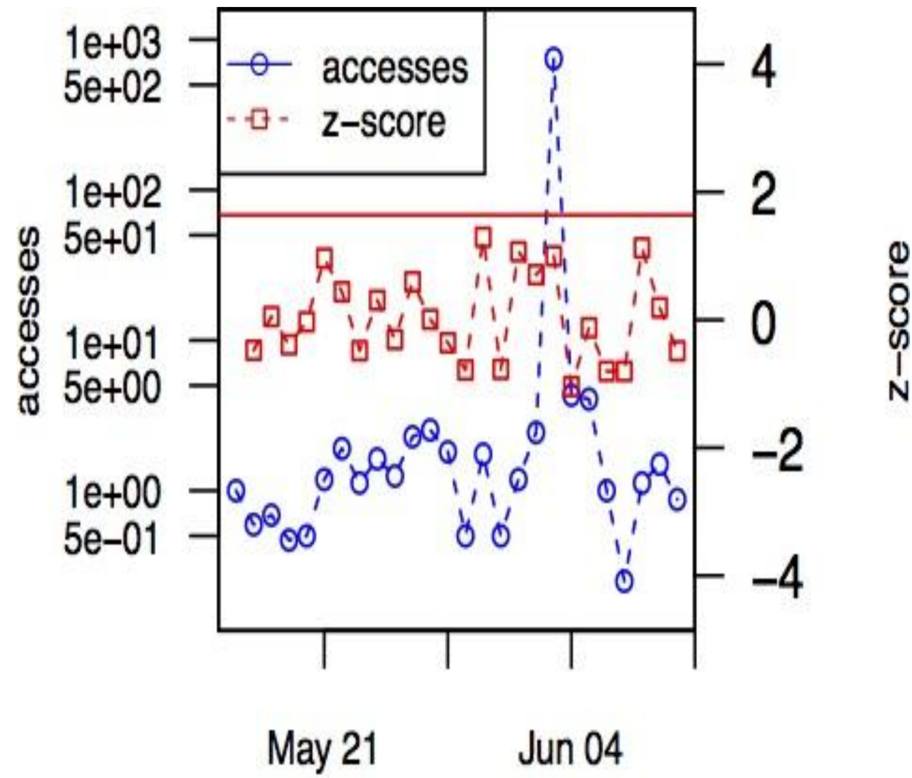
Hypothesis Testing Approach

- Flag when z-score > 1.64: red points above the red line

Data Partition A:
- Change was significant, and first flagged the day before largest increase
- Persistent change after initial spike.

Data Partition B:
- Change was not significant.
- Change did not persist after the initial spike.





Application: Bin Packing User and Application Data

- Complex optimization, taking into account many data sources and satisfying many constraints.
- The resulting number of cross-datacenter reads is one optimization criterion.
- Full logging of all (user, application) pairs would be prohibitively expensive.
- Resulting Cross-Datacenter reads can be approximated from Dapper samples.

RPC → $x = (0, 1, \ldots, 1, \ldots, 0)$

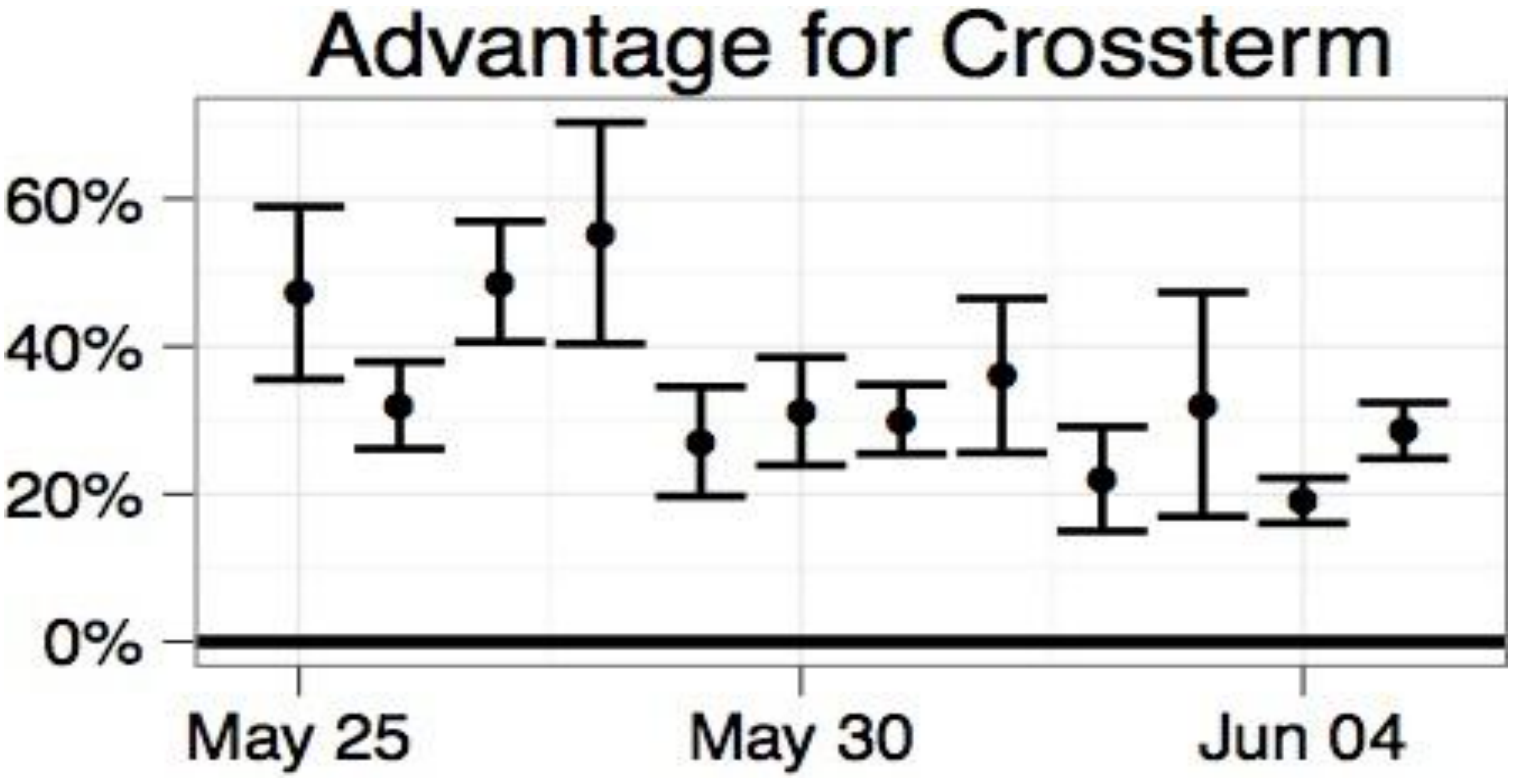
Component **j** of x will equal 1 if RPC would have caused a cross-datacenter read in the bin-packing strategy **j**

The weighted aggregation over x estimates the cross-datacenter reads for each of the of the potential bin packing strategies. When can we say that one strategy is significantly better than another in terms of cross-datacenter reads?

Two Example Strategies

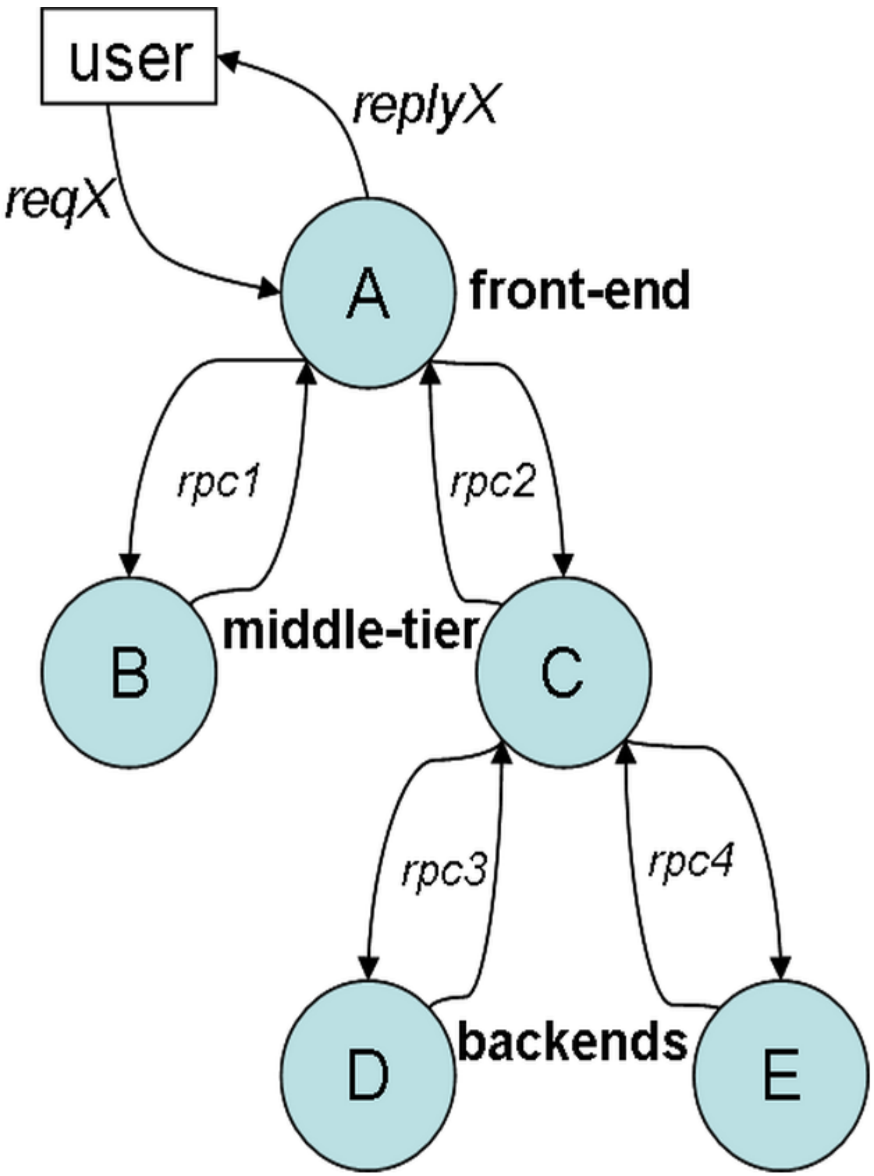
- Problem: Repack users/data in datacenters to minimize cross datacenter reads.
- Basic Strategy (First fit):
  - Fill datacenters with users/data in alphabetical order.
- Crossterm Strategy (Greedy):
  - Estimate cross user reads from training data.
  - Put pairs of users with most cross-reads in same datacenter.
- Does one consistently work better?

- Normalized difference (basic-crossterm) by the overall average of basic.
- Confidence intervals above zero means that crossterm strategy is better every day.



Dapper Sampling 2: Details

The Variance depends on the JOINT sampling probability for any two RPCs



- For every RPC that gets returned, we also know the the sampling probability **p=s*d**, which is the product of the server sampling probability and the downsampling probability.

- All RPCs in a trace share one ID, which is a uniformly generated 64-bit integer.

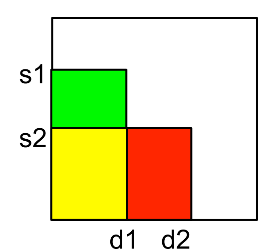
- The trace ID and its hash can be mapped to a point (s', d') on the unit square, can be modeled as a uniform draw on that square, and an RPC is returned if s' < s and d' < d.

- For RPCs in two different traces, the joint sampling probability is:

$$p_{12} = s_1 d_1 s_2 d_2$$

- For RPCs in the same trace, the joint sampling probability is:

$$p_{12} = min(s_1, s_2) \, * \, min(d_1, d_2)$$



The Math Slide: Covariance Estimate Algorithm

- GetSigmaHat returns an unbiased estimate of the covariance matrix of aggregated Dapper samples.
- Using the normal approximation, we can compute z-scores from the variance estimates.

Notes on the Algorithm

- The resulting covariance estimate is the sum of contributions over each trace.
- A valid (optional) step is to first aggregate contributions corresponding to the same values of (ID, s, d).
- While the number of RPCs within a trace may be very large, the number of distinct (s, d) values across all traces is small (< 20), so the quadratic term in Algorithm 2 is small.
- Given M distinct (s,d) combinations, and a J dimensional estimate with M and J fixed; the algorithm scales with N RPCs and T total traces as:
    - N * c_1 + T * c_2

Algorithm 1 GetSigmaHat

$M \leftarrow$ a $J \times J$ matrix of zeros.
for all $ID \in S$ **do**
  $M += ProcessSingleTrace(ID)$
end for
return M

Algorithm 2 ProcessSingleTrace

Given a collection of (s_i, d_i, x_i) corresponding to a given ID, aggregate data over the unique tuples of (s, d) to get $(s_k, d_k, \mathbf{y}_k)$ where $\mathbf{y}_k = \sum_{\{j|(s_j,d_j)=(s_k,d_k)\}} \mathbf{x}_j$ and we let K_t be the number of distinct tuples resulting form this aggregation.

$M \leftarrow$ a $J \times J$ matrix of zeros.

for all $k \in 1 : K_t$ **do**
  for all $k' \in 1 : K_t$ **do**
    $w = \frac{1 - \max(s_k, s_{k'}) * \max(d_k, d_{k'})}{s_k s_{k'} d_k d_{k'}}$
    $M += w * (y_k \otimes y_{k'})$
  end for
end for
return M

Algorithm Scalability

- Compute joint probabilities for pairs of RPCs
- Compute variance in estimates from joint probabilities

- Complexity: Linear with number of traces
- Quadratic in number of (server sampling,down-sampling) probabilities, but that is usually small

Conclusions

- Aggregated Dapper samples are useful when direct measurements are not available.
- A detailed understanding of the sampling mechanisms is required to estimate the variance of the estimate.
- Using variance estimates allows us to reliably compare different aggregates, e.g.:
    - When a detected change in IO rates is real (compare rates for different days)
    - Select bin-packing strategies (compare cross-datacenter read estimates)