Each user request touches hundreds of systems
The workload is **constantly** evolving

Many products

Growing user base

Rapid software change

- Facebook: 3 daily releases
- Instagram: cont. release
Goals

• How many machines does each software system need?

• Can we serve peak load?

• Are we operating efficiently?
Common approaches to capacity management

- **Load modeling**: simulate how system behaves at high load
- **Load testing**: benchmark using synthetic workloads
Live user traffic is the most representative workload

- Accurate distribution of reads & writes
- Do not need a custom test setup
Live traffic load tests measure **peak serving capacity**

- Direct live user traffic at target
Live traffic load tests measure **peak serving capacity safely**

- Monitor health metrics
  - Response latency
  - Server error
- Reset load when thresholds are hit
Roadmap

Kraken

• Kraken measures peak serving capacity at all scales
  • A single web server
  • A single cluster
  • An entire geographical region

• Kraken identifies bottlenecks limiting utilization
  • Load imbalance
  • Network saturation

• Challenges in deploying Kraken
Kraken uses weights to route requests

Edge weight

Cluster weight

Frontend Cluster

Region

Region

Region

Web server weight

Backend Cluster

Service Cluster

Newsfeed

Search
Kraken measures a web server’s peak serving capacity

- Peak web server capacity: 175 requests per second (RPS)
- Production target: 90% utilization i.e., 157 RPS
Kraken measures a cluster’s peak serving capacity

- Max cluster capacity = (web server capacity) * (num. web servers in cluster)
Kraken measures a region’s peak serving capacity

- We now serve 20% more users with the same infrastructure
Inefficient load balancing limits utilization
Network saturation limits utilization
Challenge: non-linear response to traffic shifts
Challenge: how can we foster experimentation?

- Set conservative thresholds
- Communicate widely about tests
- Encourage collaboration
  - Monitoring
  - Failure mitigation strategies
Conclusion

• We’ve run 50+ regional, 1000+ cluster live traffic load tests in 3 years

• Kraken has helped us identify hundreds of bottlenecks and verify fixes

• We can now serve 20% more users with the same infrastructure
Kraken: assumptions and caveats

• Stateless servers

• Routable requests

• Load impacts downstream systems
Kraken: user traffic management
Kraken: live traffic load tests
# Health metrics for systems affected by web load

<table>
<thead>
<tr>
<th>Service type</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web servers</td>
<td>CPU utilization, latency, error rate, fraction of operational servers</td>
</tr>
<tr>
<td>Aggregator–leaf</td>
<td>CPU utilization, error rate, response quality</td>
</tr>
<tr>
<td>Proxygen software L7 load balancer</td>
<td>CPU utilization, latency, connections, retransmit rate, Ethernet utilization, memory capacity utilization</td>
</tr>
<tr>
<td>Memcache</td>
<td>Latency, object lease count</td>
</tr>
<tr>
<td>TAO</td>
<td>CPU utilization, write success rate, read latency</td>
</tr>
<tr>
<td>Batch processor</td>
<td>Queue length, exception rate</td>
</tr>
<tr>
<td>Logging</td>
<td>Error rate</td>
</tr>
<tr>
<td>Search</td>
<td>CPU utilization</td>
</tr>
<tr>
<td>Service discovery</td>
<td>CPU utilization</td>
</tr>
<tr>
<td>Message delivery</td>
<td>CPU utilization</td>
</tr>
</tbody>
</table>
Continuous runs measuring a web server’s capacity
Some lessons from a thousand Kraken tests

• Simplicity is key to Kraken’s success.

• Identifying the right performance, error rate and latency metrics to track is difficult.

• Cheap solutions, like allocating capacity or fixing misconfiguration, are often more impactful than profile-based tuning or system redesign.