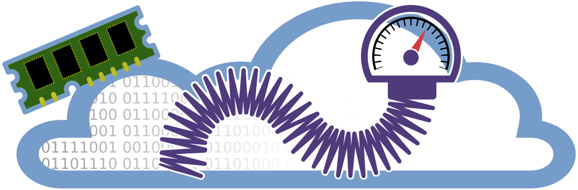


Memory Elasticity Benchmark

Liran Funaro Orna Agmon Ben-Yehuda Assaf Schuster

Department of Computer Science

TECHNION
Israel Institute of Technology

SYSTOR'2020

Improve Utilization

Improve Utilization

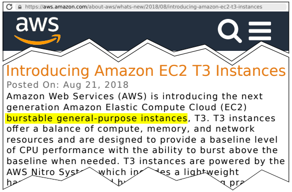
- Cloud providers aim to make more money off the same hardware

Improve Utilization

- Cloud providers aim to make more money off the same hardware
- Rigid allocation prevents optimal resource utilization

  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Stochastic Resource Allocation". In: *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '19)*. USENIX Association. Providence, RI, USA: ACM, 2019. ISBN: 978-1-4503-6020-3/19/04

Elastic Allocation


https://aws.amazon.com/about-aws/whats-new/2018/08/introducing-amazon-ec2-t3-instances

aws

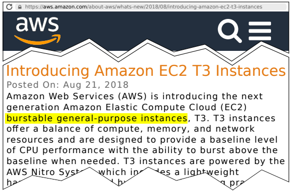
Introducing Amazon EC2 T3 Instances

Posted On: Aug 21, 2018

Amazon Web Services (AWS) is introducing the next generation Amazon Elastic Compute Cloud (EC2) burstable general-purpose instances, T3. T3 instances offer a balance of compute, memory, and network resources and are designed to provide a baseline level of CPU performance with the ability to burst above the baseline when needed. T3 instances are powered by the AWS Nitro Syst... which in...es a lightweight

- Burstable performance offers CPU elasticity
- Clients can "burst" to a higher level when required
  - Allow changing resource consumption on the fly
  - Exploiting resources that are momentarily unused by others

Elastic Allocation



https://aws.amazon.com/about-aws/whats-new/2018/08/introducing-amazon-ec2-t3-instances

Introducing Amazon EC2 T3 Instances

Posted On: Aug 21, 2018

Amazon Web Services (AWS) is introducing the next generation Amazon Elastic Compute Cloud (EC2) burstable general-purpose instances, T3. T3 instances offer a balance of compute, memory, and network resources and are designed to provide a baseline level of CPU performance with the ability to burst above the baseline when needed. T3 instances are powered by the AWS Nitro Syst... which in...es a lightweight

- Burstable performance offers CPU elasticity
- Clients can "burst" to a higher level when required
  - Allow changing resource consumption on the fly
  - Exploiting resources that are momentarily unused by others

- More clients can be allocated to the same physical servers

Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Stochastic Resource Allocation". In: *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '19)*. USENIX Association. Providence, RI, USA: ACM, 2019. ISBN: 978-1-4503-6020-3/19/04

Memory is the New Bottleneck

- Memory is the new bottleneck
  - It is an expensive resource that limits machine occupancy

Memory is the New Bottleneck

- Memory is the new bottleneck
  - It is an expensive resource that limits machine occupancy
- Memory elasticity schemes should be a natural extension to CPU elasticity
  - Allowing clients to use more memory in the same VM/container than their initial memory allocation

Memory Elastic Applications

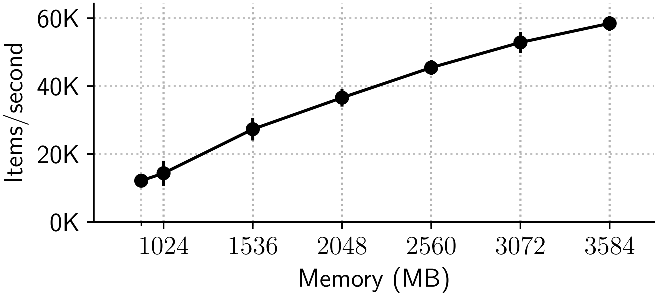
- Applications that can burst

Memory Elastic Applications

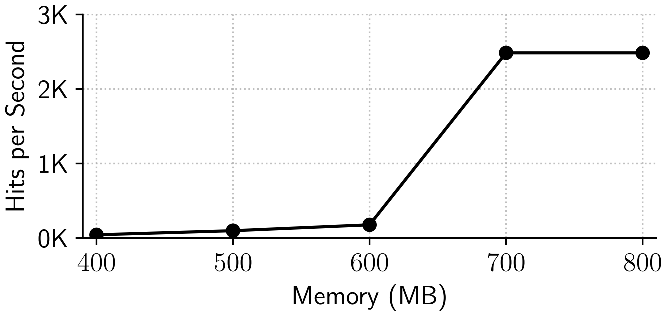
- Applications that can burst

Memory Elastic Applications

- Applications that can burst
- Whose performance is proportional to their memory usage



Memory Elastic Applications Exists?



- Memory-elastic applications are scarce
  - Maximal memory footprint is dictated by the current application workload
- The OS's swapping allows seamless application operation
  - Even a minor memory loss may degrade the performance significantly

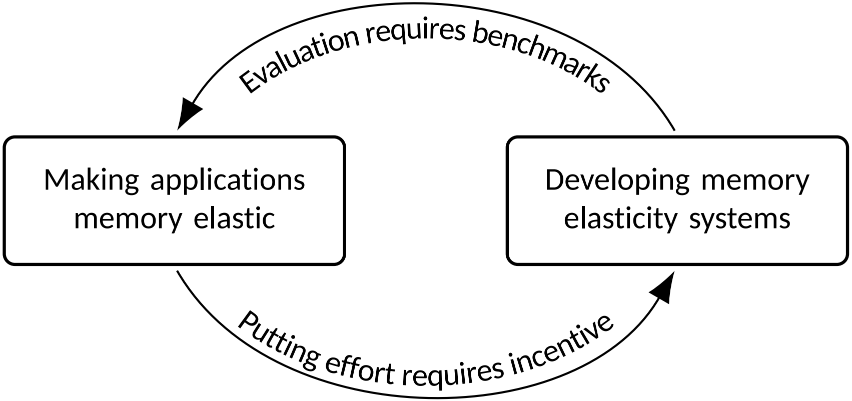
Where are the Memory Elastic Applications?

- Why most applications can scale with CPU? But not for memory?

Where are the Memory Elastic Applications?

- Why most applications can scale with CPU?
  But not for memory?
- Multi-core architectures and CPU schedulers were the incentive

Circular Dependency



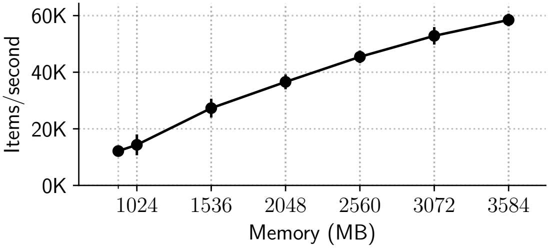
Applications with Resource Trade-off

- Mechanisms that were designed to allow trade-off between memory and other resources can be used to provide memory elasticity

Memory as Cache

Applications that use the RAM to cache computation results, network traffic, and so on (e.g., using Memcached)

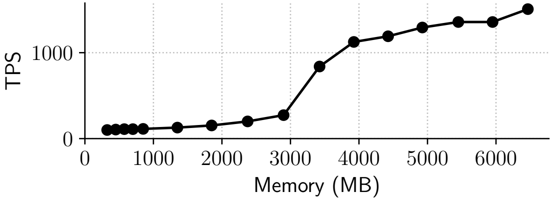
- Improve cache hit-rate when more memory is available to the operating system



Intermediate Buffers

Applications that use intermediate buffers (e.g., Hadoop, Spark)

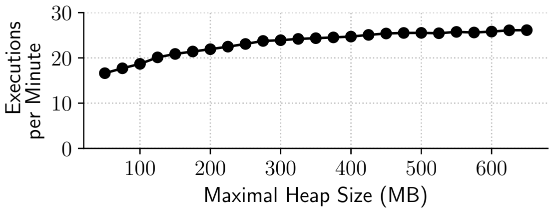
- Can use larger memory buffers to reduce disk access and speed up temporarily data-heavy operations
  - E.g., sorting and large matrix multiplication



Garbage Collected Memory

Applications with automatic memory management (e.g., Java applications)

- May need fewer garbage-collection cycles with a larger heap, and improve their performance



Multiple Short-Lived Jobs

Applications that have multiple short-lived jobs, each with different memory requirements (e.g., Nginx)

- Web servers might require a certain memory to handle each session
- They may be able to handle more concurrent sessions when more memory is available

Memory-Aware Applications

- *Memory-aware* applications adjust their memory consumption according to the available memory observed during their initiation period
  - But cannot adjust it during runtime
- Most of the commonly used memory trade-offs we mentioned are predefined and implemented as memory-aware applications

Memory-Aware Applications

- *Memory-aware* applications adjust their memory consumption according to the available memory observed during their initiation period
  - But cannot adjust it during runtime
- Most of the commonly used memory trade-offs we mentioned are predefined and implemented as memory-aware applications
- Can be made memory-elastic by restarting them when the memory changes
  - Not suitable when the application needs to be continuously available

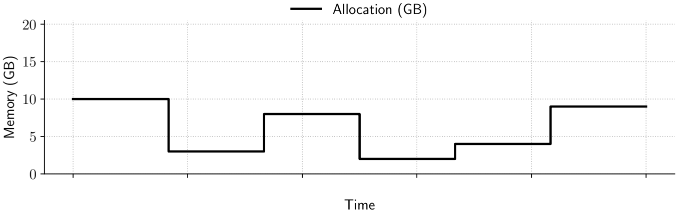
Memory-Aware Applications

- *Memory-aware* applications adjust their memory consumption according to the available memory observed during their initiation period
  - But cannot adjust it during runtime
- Most of the commonly used memory trade-offs we mentioned are predefined and implemented as memory-aware applications
- Can be made memory-elastic by restarting them when the memory changes
  - Not suitable when the application needs to be continuously available
- With a small effort, these applications can be tweaked to become memory-elastic

Tweaked Memcached

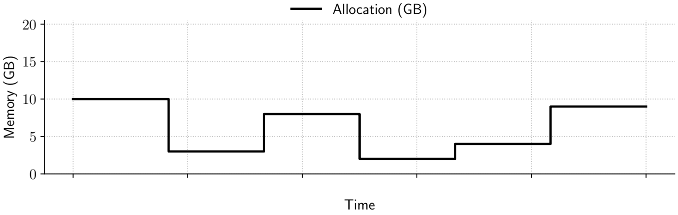
Elastic memcached supports changing its memory footprint upon receiving a command via a socket

Naive Metric



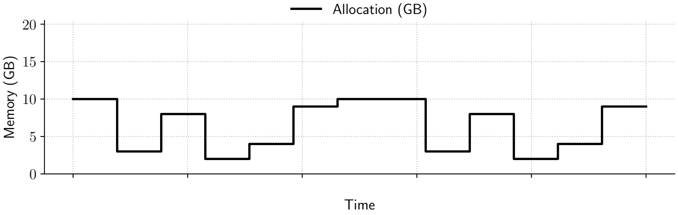
- Compare the performance of two applications under the same dynamic memory conditions and consider the one with the better results as more memory-elastic

Naive Metric



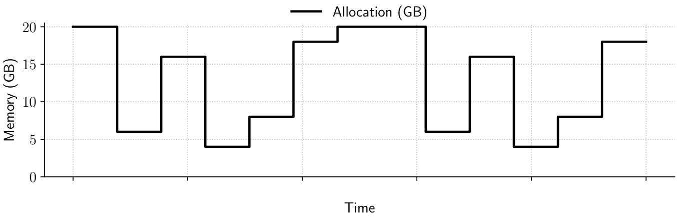
- Compare the performance of two applications under the same dynamic memory conditions and consider the one with the better results as more memory-elastic
- The results may be sensitive to the order or frequency of memory allocations

Naive Metric



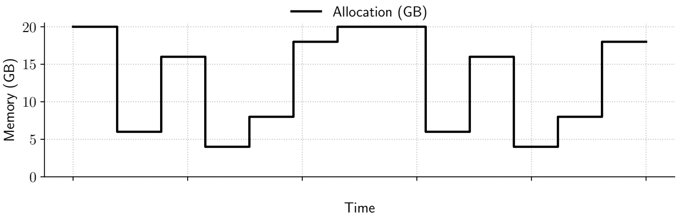
- Compare the performance of two applications under the same dynamic memory conditions and consider the one with the better results as more memory-elastic
- The results may be sensitive to the order or frequency of memory allocations

Naive Metric



- Compare the performance of two applications under the same dynamic memory conditions and consider the one with the better results as more memory-elastic
- The results may be sensitive to the order or frequency of memory allocations

Naive Metric



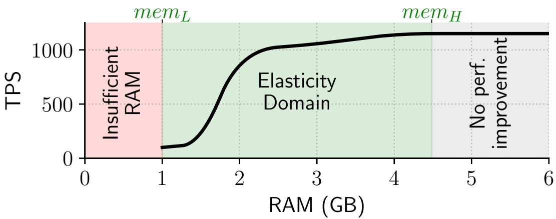
- Compare the performance of two applications under the same dynamic memory conditions and consider the one with the better results as more memory-elastic
- The results may be sensitive to the order or frequency of memory allocations
- This is because we try to infer memory elasticity from observations of metrics that only hint about elasticity, but do not measure it directly

Our Goal

- Our goal is
  - To quantify an application's behavior in a dynamic memory scenario
  - To compare it to other applications
  - Using metrics that directly relate to memory elasticity

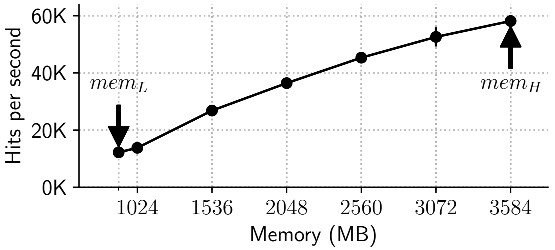
Static Metrics

- Static memory→performance function (P_{mem}) that describes the performance of the application given a static memory allocation
- *Elasticity domain*: $[mem_L, mem_H]$
- *Elasticity range*: $mem_H - mem_L$

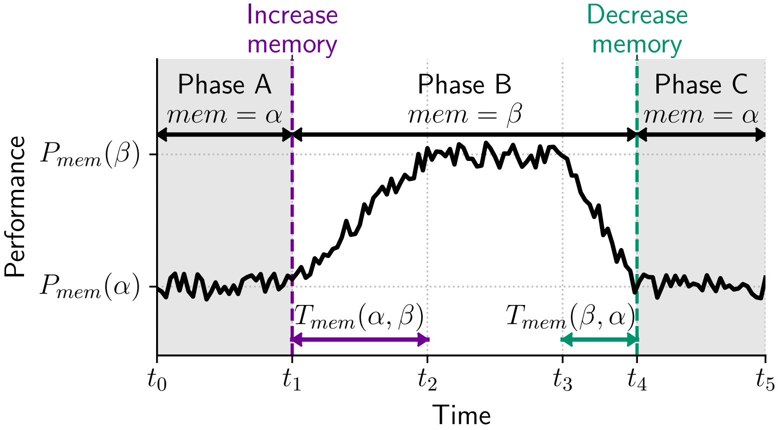


Memcached Static Metrics

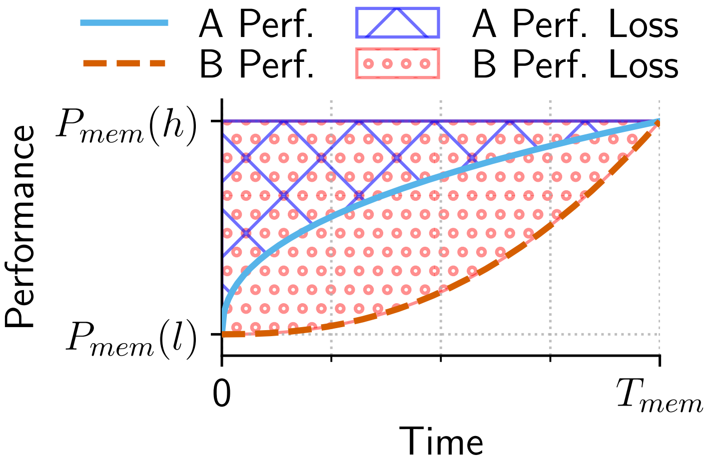
- Elasticity domain: from 1 GB to 3.5 GB
- Elasticity range: 2.5 GB



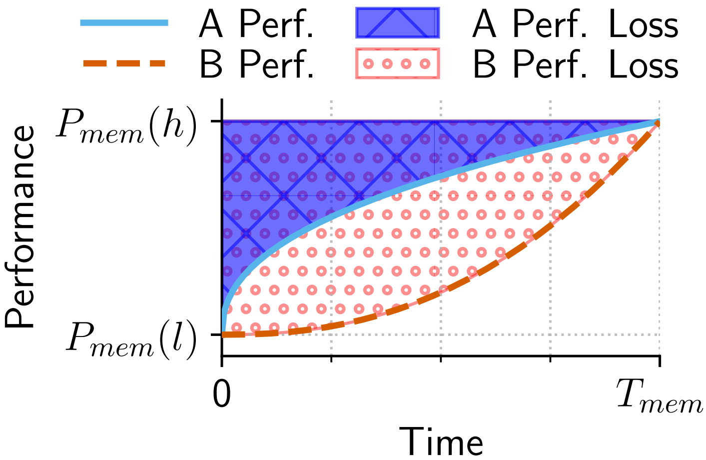
Dynamic Metrics



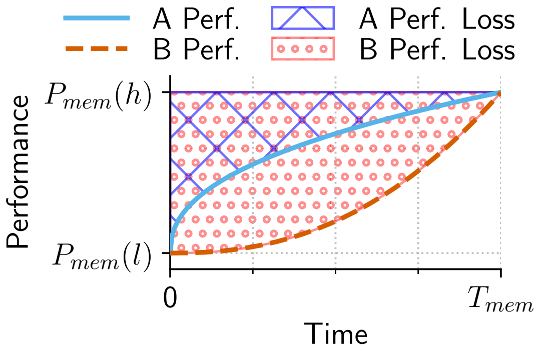
Performance Loss During the Transient Period (1)

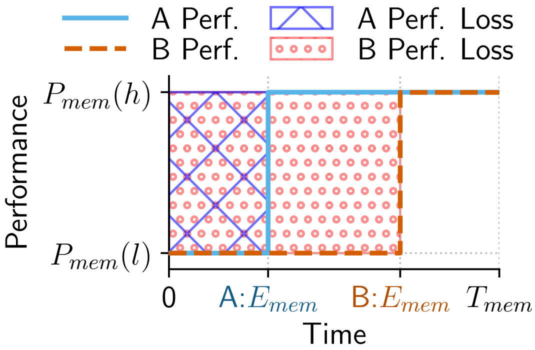


Performance Loss During the Transient Period (1)

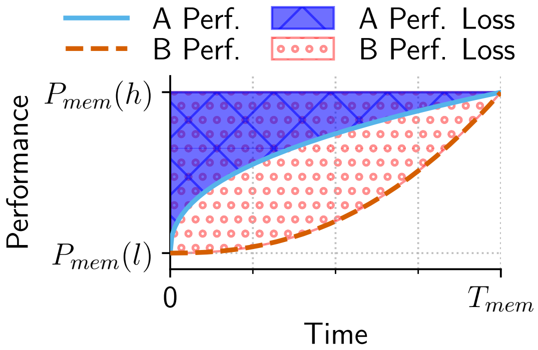


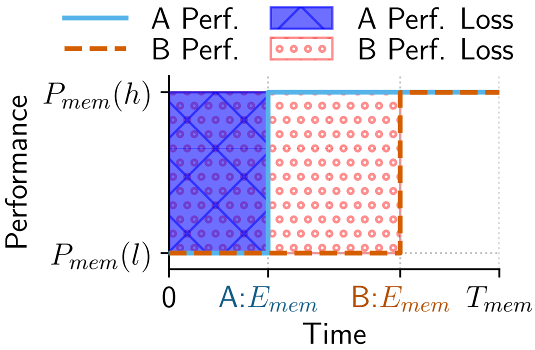
Performance Loss During the Transient Period (2)



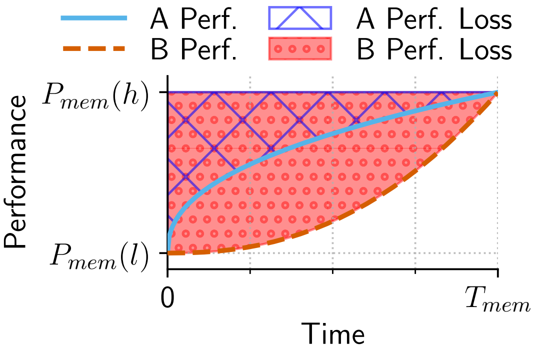


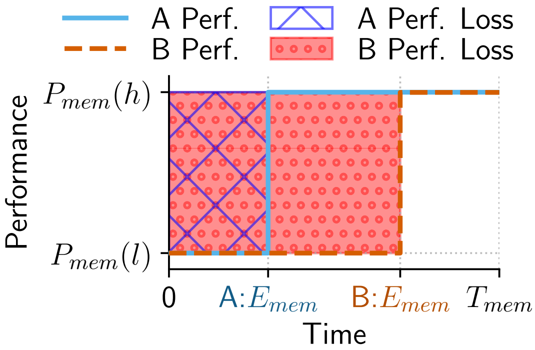
Performance Loss During the Transient Period (2)



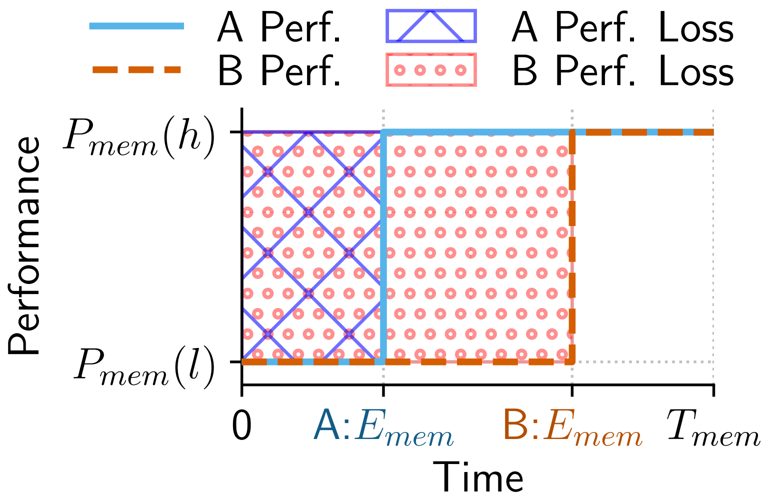


Performance Loss During the Transient Period (2)

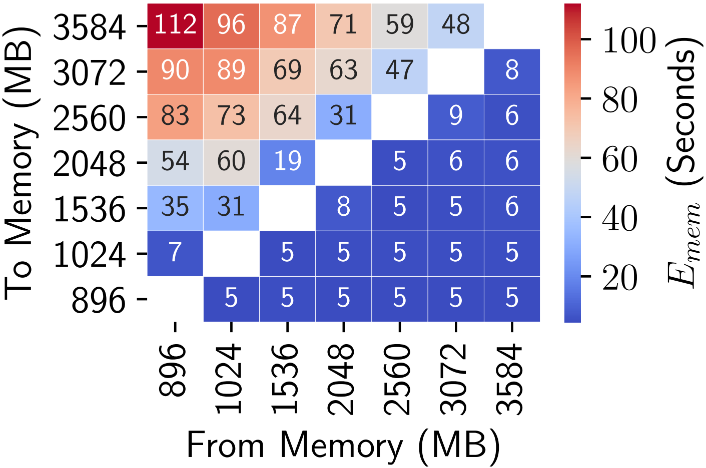




Performance Loss During the Transient Period (3)

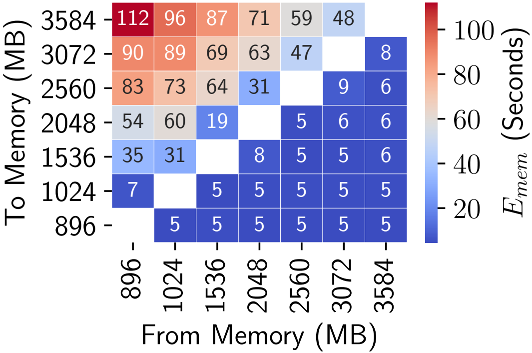


Elastic Memcached Average Measured E_{mem}

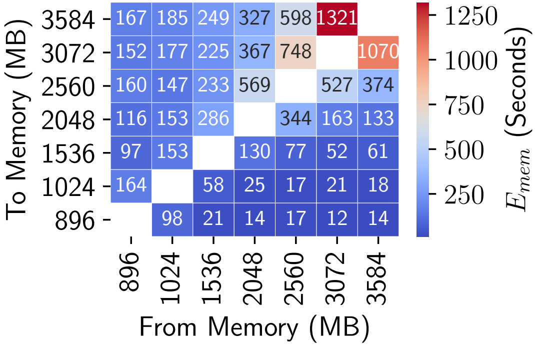


Elastic vs. Off-the-shelf Memcached

Elastic



Off-the-shelf



Conclusions

- We showed a few major building blocks that can be made memory elastic
  - Cache, intermediate buffers, garbage-collection and schedulers
- We defined metrics that are comparable across applications
  - Elasticity range and E_{mem}
- We defined characteristics that can be used by clients to configure their virtual machine and their application in a memory elastic cloud environment
  - P_{mem} and T_{mem}
- Our framework is available from `github.com/liran-funaro/elastic-benchmarks`

Liran Funaro: `funaro@cs.technion.ac.il`