Jumanji: The Case for Dynamic NUCA in the Datacenter

Brian C. Schwedock, Nathan Beckmann

MICRO 2020
Datacenters care about security and tail latency

- **Security**: data and performance protection among untrusted users (e.g., VMs)
- **Tail latency**: execution time of slowest application requests

*Computer systems must be redesigned to efficiently enforce these goals!*
Prior D-NUCAs do not work for datacenter applications!

- D-NUCAs improve energy efficiency > 50%!
- **Problem**: After 20 years of research, all D-NUCAs care only about data movement, making them unreliable for datacenter apps.

Jumanji is a new D-NUCA that improves security, tail latency, *and* energy efficiency!
High-level overview of Jumanji

1. Reserves space to meet tail-latency deadlines
2. Isolates VMs across banks to defend against LLC attacks
3. Optimizes batch data placement within each VM

Jumanji: places apps’ data in the LLC to meet apps’ high-level goals.
Agenda

● Motivation
  o Security
  o Tail latency
  o Prior D-NUCAs

● Jumanji’s design

● Evaluation

● Conclusion
LLC has a big impact on security and tail latency

- Data movement within the LLC exposes side-channel attacks and determines tail latencies
- Many recently discovered side channels occur at the LLC [1]
- Larger LLC allocations greatly reduce tail latency [2,3,4]
- These works ignore NUCA, and by doing so, miss additional security vulnerabilities and harm efficiency

[2] Chen et al., PARTIES, ASPLOS 2019
[4] Lo et al., Heracles, ISCA 2015
Prior work

**Security**: sharing LLC banks is unsafe

**Tail latency**: ignoring NUCA wastes cache space

**D-NUCA**: ignoring application goals is harmful
Prior LLC designs are **insecure** and **inefficient**

- Prior LLC designs focus on defending conflict attacks (e.g., prime + probe); **way-partitioning** is the most common defense
- **Insecure**: Limited LLC associativity prevents defending all processes
- **Insecure**: We demonstrate new **port** and **replacement-policy** attacks on prior designs
- **Wasteful**: Ignoring NUCA $\rightarrow$ lots of unnecessary data movement
Demonstration: Sharing LLC banks is unsafe

Port attack on Xeon processor LLC

Attacker detects victim accessing target bank!
Only prior LLC defense: IRONHIDE

- IRONHIDE is the only prior solution which defends all LLC attacks
- It isolates applications across LLC banks, creating two regions: secure and insecure
- Although it also defends non-LLC attacks, it only supports one secure region at a time
- Additionally, IRONHIDE does not address tail latency and does not minimize data movement as well as D-NUCA
Prior work

**Security:** sharing LLC banks is unsafe

**Tail latency:** ignoring NUCA wastes cache space

**D-NUCA:** ignoring application goals is harmful
Prior LLC designs for **tail latency** are inefficient

- Prior LLC designs for tail latency dynamically allocate cache space, but ignore NUCA
- **Wasteful:** Ignoring NUCA $\Rightarrow$ lots of unnecessary data movement for latency-critical applications
- **Wasteful:** Latency-critical applications thus need more cache space to meet deadlines, *harming co-running batch applications*
- (And are still insecure)
Prior LLC designs for **tail latency** are inefficient

With S-NUCA, LLC accesses have high latency!

Placing data closer lowers avg access latency...

... so less space is needed to maintain tail latency

Which leaves more space for co-running batch applications to improve throughput

D-NUCA meets tail-latency deadlines much more efficiently!
D-NUCA meets deadlines with less LLC space

- Simulated 20-core CMP
- Running latency-critical application *Xapian* in isolation
- Measured tail latency with different allocation sizes using way-partitioning (S-NUCA) and nearby data placement (D-NUCA)

![Graph showing latency comparison between S-NUCA and D-NUCA](image)

- S-NUCA requires 75% more space!
Prior work

**Security:** sharing LLC banks is unsafe

**Tail latency:** ignoring NUCA wastes cache space

**D-NUCA:** ignoring application goals is harmful
Unfortunately, prior D-NUCAs fail in the datacenter

- Dynamic non-uniform cache access (D-NUCA) architectures place data in LLC banks to **minimize data movement**
- … but data movement != security and tail latency

Jigsaw performs well on throughput-oriented batch applications, but poorly for all other goals.

Beckmann et al., Jigsaw, PACT 2013
Jumanji is the solution!

Defends all applications against more LLC attacks

Meets tail-latency deadlines with minimal data movement

Maximizes throughput of co-running batch applications

Simple design and small software changes to Jigsaw
Agenda

● Motivation
  o Security
  o Tail latency
  o Prior D-NUCAs

● Jumanji’s design

● Evaluation

● Conclusion
Jumanji Software

1. Reserve lat-crit allocation
2. Partition banks among VMs
3. Optimize batch within each VM

USER SPACE

VM 1

VM 2

Request completes

100 ms Periodically trigger reconfiguration

Feedback controller

Jumanji Hardware

Miss curve monitors

VTB (update data placement)
Step 1: Meeting tail-latency deadlines

- Previous allocation
- Tail latency of recent requests
- User-defined deadline

**Feedback controller**

- Tail < 85% * deadline: Decrease by 10%
- Tail < 95% * deadline: No change
- Tail < 105% * deadline: Increase by 10%
- Tail >= 105% * deadline: Boost to “safe” size

100 ms Periodically trigger reconfiguration

Greedily place data near thread(s)
Step 2: Defending LLC attacks

Merged from each VM’s batch-app curves

Per-VM miss curves

Modified to allocate at bank granularity to prevent VMs from sharing banks. **This is how Jumanji defends LLC bank attacks.**

Qureshi et al., UCP, MICRO 2009
Mukkara et al., Whirlpool, ASPLOS 2016
Step 3: Optimizing for batch performance

Execute Jigsaw on batch apps within each VM

Beckmann et al., Jigsaw, PACT 2013
Jumanji Hardware (borrowed from Jigsaw)

Out-of-order core

L1d L1i TLB

L2

VTB

UMON

20-core CMP

VC id: 42

Modified: TLB
Added: VTB, UMON

VTB

<table>
<thead>
<tr>
<th>VC ids</th>
<th>Placement descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2 1 2 0 0 ... 1 1</td>
</tr>
</tbody>
</table>
| 42     | 0 1 0 0 2 ... 0 2  

Addr: 0xBEEF

Hash

bank id

0xBEEF \(\rightarrow\) LLC bank 1

Beckmann et al., Jigsaw, PACT 2013

Beckmann et al., CDCS, HPCA 2015
Agenda

● Motivation
  o Security
  o Tail latency
  o Prior D-NUCAs

● Jumanji’s design

● Evaluation

● Conclusion
Evaluation Methodology

- Simulate 20-core CMP based on Nehalem using ZSim
- 20 single-threaded applications split into 4 VMs
- Each VM has
  - 1 latency-critical app (from Tailbench)
  - 4 batch apps (from SPEC CPU2006)
- Latency-critical workloads
  - 4 copies of the same latency-critical app
  - Random mixes of latency-critical apps
- Batch workloads
  - 40 random mixes of batch apps for each latency-critical workload
Evaluation Methodology – LLC Designs

Adaptive:
Meets deadlines using way-partitioning and feedback controller

VM-Part:
Also way-partitions across VMs to defend conflict attacks (but not our new attacks)

Jigsaw

Jumanji
Jumanji meets deadlines and speeds up batch apps

Latency-critical apps: Xapian x 4

- Adaptive
- VM-Part
- Jigsaw
- Jumanji

Up to 465

Normalized Tail Latency vs. Latency App

Weighted Speedup % vs. Workload Mix
Jumanji meets deadlines and speeds up batch apps

Latency-critical apps: random mixes from Tailbench
See the paper for more results!

- Jumanji’s data placement is nearly ideal
- Jumanji scales well with VM size

- Also...
  - Energy savings
  - Security analysis
  - System sensitivity study
Agenda

● Motivation
  ○ Security
  ○ Tail latency
  ○ Prior D-NUCAs

● Jumanji’s design

● Evaluation

● Conclusion
Jumanji makes D-NUCA viable in the datacenter

- Jumanji recognizes the advantages D-NUCA provides for security and tail latency
- Isolating untrusted VMs across LLC banks provides stronger security than prior designs
- Placing latency-critical data near cores saves cache space for co-running batch applications
- The overall design makes D-NUCA work for modern application goals

Brian Schwedock bschwedo@andrew.cmu.edu
Nathan Beckmann beckmann@cs.cmu.edu
This presentation and recording belong to the authors. No distribution is allowed without the authors’ permission.