Challenges and Opportunities for Systems Using CXL Memory

Emmett Witchel, UT Austin
Credit to many, including
Zhiting Zhu, Newton Ni, Yibo Huang, Zhipeng Jia (Google), Yan Sun (UIUC), Nam Sung Kim (UIUC)

Once

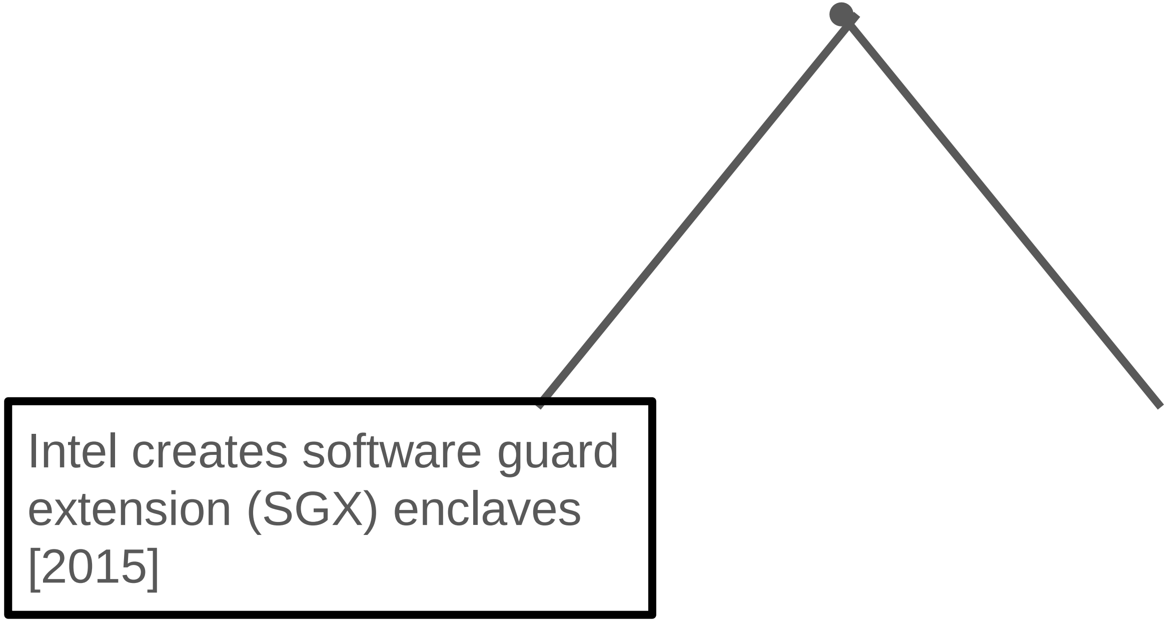
upon a

time

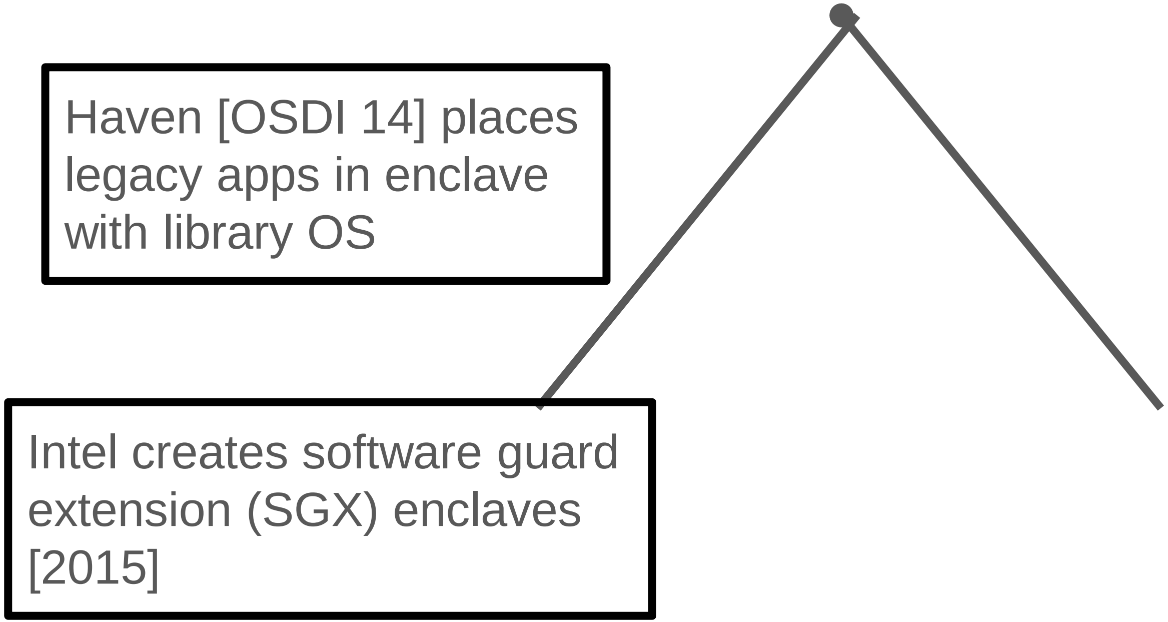
In

the

beginning



Intel creates software guard extension (SGX) enclaves [2015]



Haven [OSDI 14] places legacy apps in enclave with library OS

Intel creates software guard extension (SGX) enclaves [2015]

from
ImageFX (Google)

Intel creates software guard extension (SGX) enclaves [2015]

from
ImageFX (Google)

Panoply [NDSS 17], Komodo [SOSP 17], Occlum [ASPLOS 20], minimize TCB in enclave

Intel creates software guard extension (SGX) enclaves [2015]

from
ImageFX (Google)

Panoply [NDSS 17], Komodo [SOSP 17], Occlum [ASPLOS 20], minimize TCB in enclave

Intel creates software guard extension (SGX) enclaves [2015]

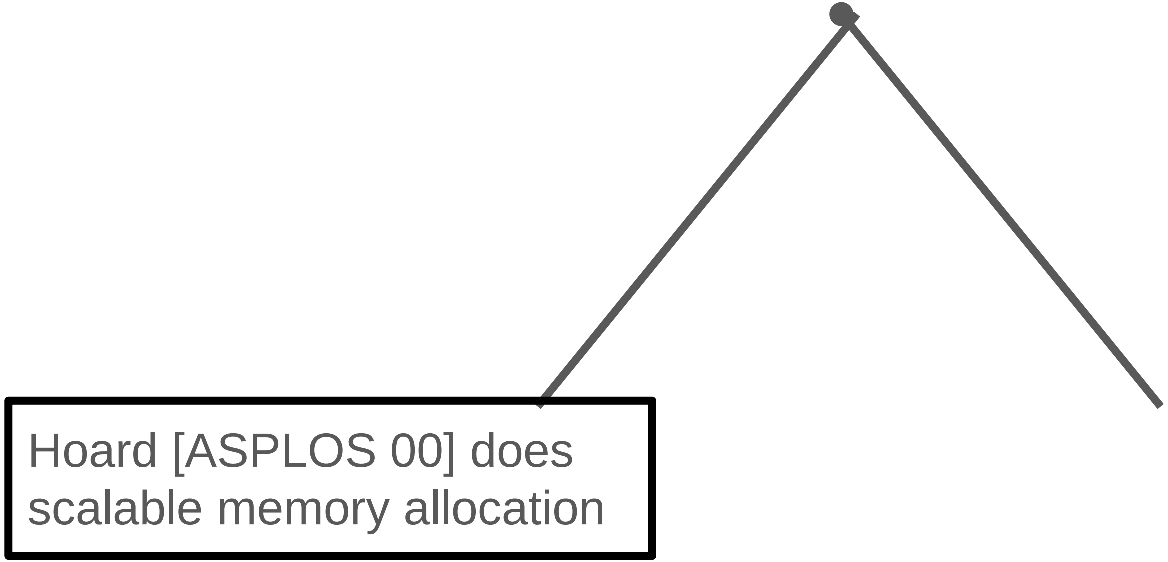
Intel makes trust domain extensions (TDX) to secure VMs [2023]

from
ImageFX (Google)

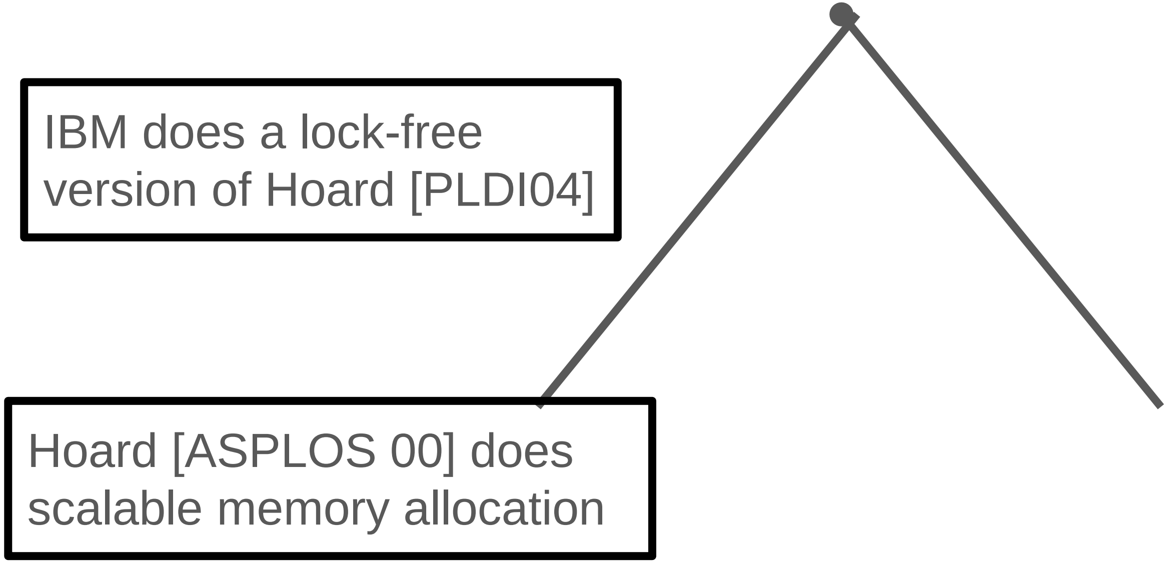
P
[S
2

Intel creates software guard extension (SGX) enclaves [2015]

Intel makes trust domain extensions (TDX) to secure VMs [2023]

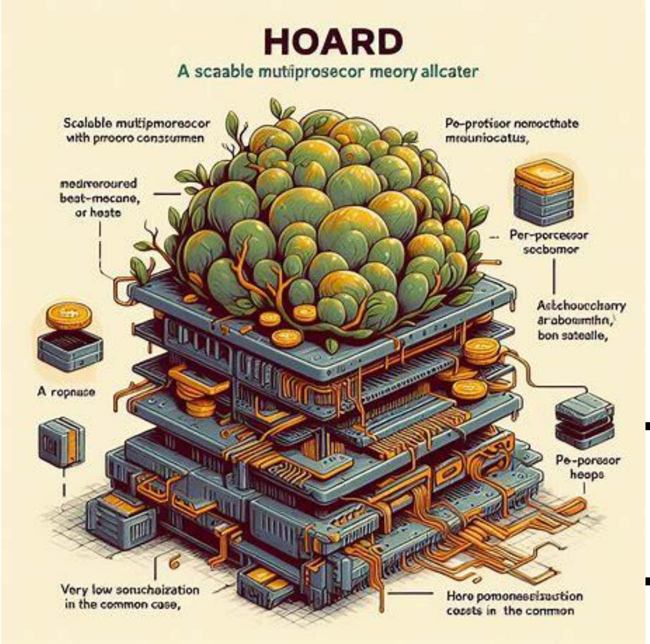


Hoard [ASPLOS 00] does scalable memory allocation



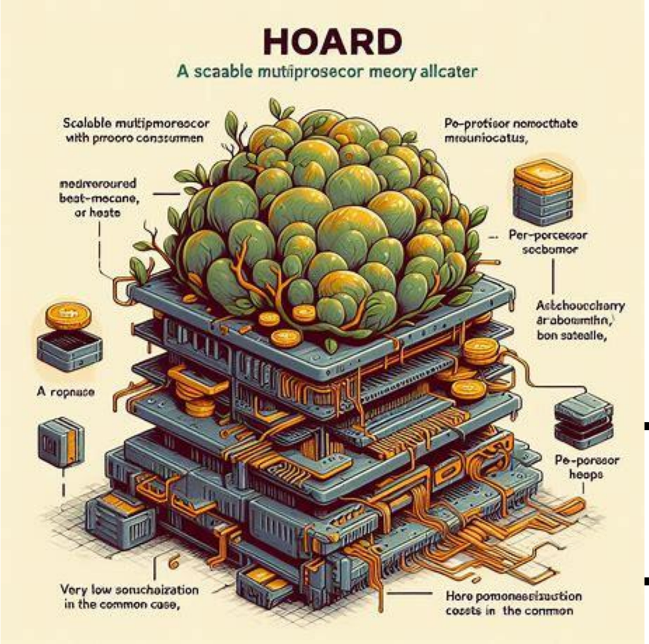
IBM does a lock-free version of Hoard [PLDI04]

Hoard [ASPLOS 00] does scalable memory allocation


HOARD
A scaable mutiprosecor meory allcater
Scalable multipmoreocor with prooro consumen
Po-protisor nomocthate mrounioocatus,
nodrmoroured beat-mocane, or heste
Per-porcesor socbomor
Astchoucchory arabommthn, bon soteolle,
A ropnace
Pe-poresor heaps
Very low sonchaization in the common case,
Horo pomoneesizaction cossts in the conmon

from
Copilot (MS)

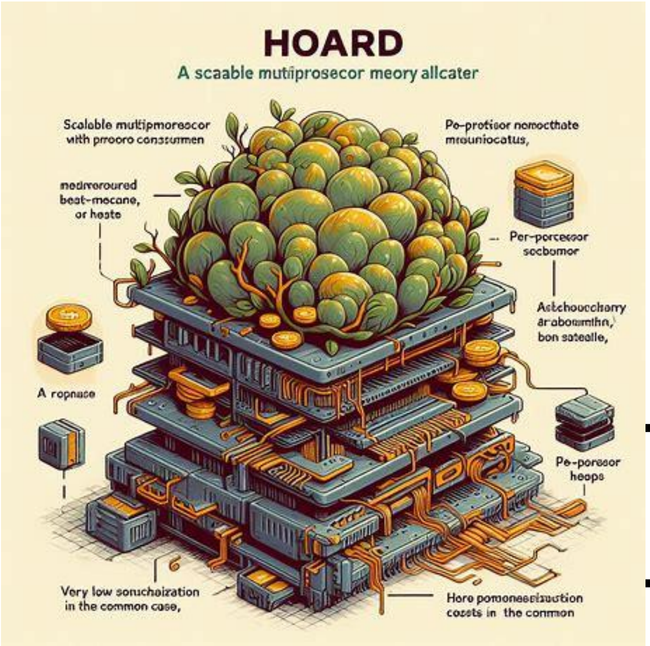
Hoard [ASPLOS 00] does scalable memory allocation


HOARD
A scaable mutiprosecor meory allcater
Scalable multipmorescor with prooro consumen
Po-protisor nomocthate mrauniocatus,
nodrerourod beat-mecane, or heate
Per-porcesoor socbomor
Astchoucchorry arabommthn, bon satealle,
A ropnase
Pe-porosor heaps
Very low sonuchaization in the common case,
Horo pomoneesizuction cosets in the conmon

from
Copilot (MS)

Apple adopts Hoard, cites ASPLOS paper in code [08]

Hoard [ASPLOS 00] does scalable memory allocation

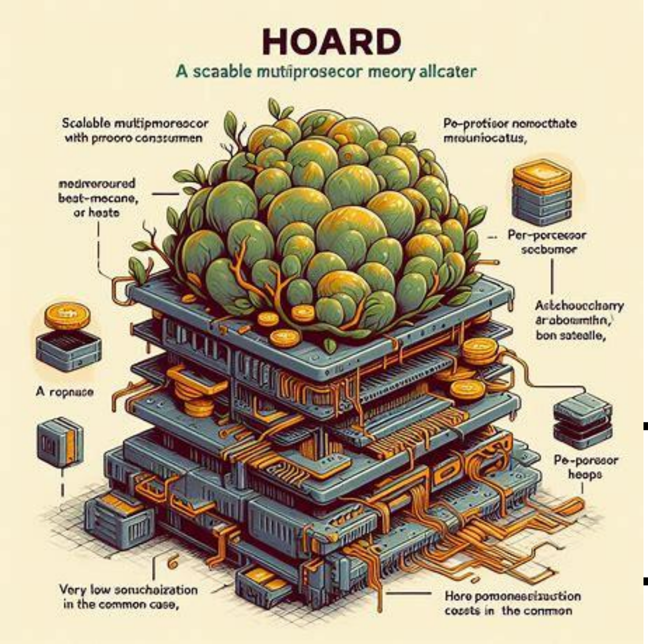

HOARD
A scaable mutiprosecor meory allcater
Scalable mutipmoreocor with prooro consumen
Po-protisor nomocthate mrouniocatus,
medreroured beat-mecane, or heate
Per-porcesor socbomor
Astchoucchorry arabomnthn, bon satealle,
A ropnace
Pe-porssor heaps
Very low sonschaization in the common case,
Horo pomonecisuction cosets in the conmon

from
Copilot (MS)

Apple adopts Hoard, cites ASPLOS paper in code [08]

Hoard [ASPLOS 00] does scalable memory allocation

LLAMA [ASPLOS 20] uses machine learning to reduce fragmentation


HOARD
A scaable mutiprosecor meory allcater
Scalable multipmorescor with prrooro consumen
Pe-protisor nomocthate mrauniocatus,
nodmeroured beat-mecane, or heate
Per-porcessor socbomor
Astchoocchorry arabommthn, bon sateallle,
A ropnase
Pe-porasor heaps
Very low sonuchaization in the common case,
Horo pomoneasisuction cossts in the conmon

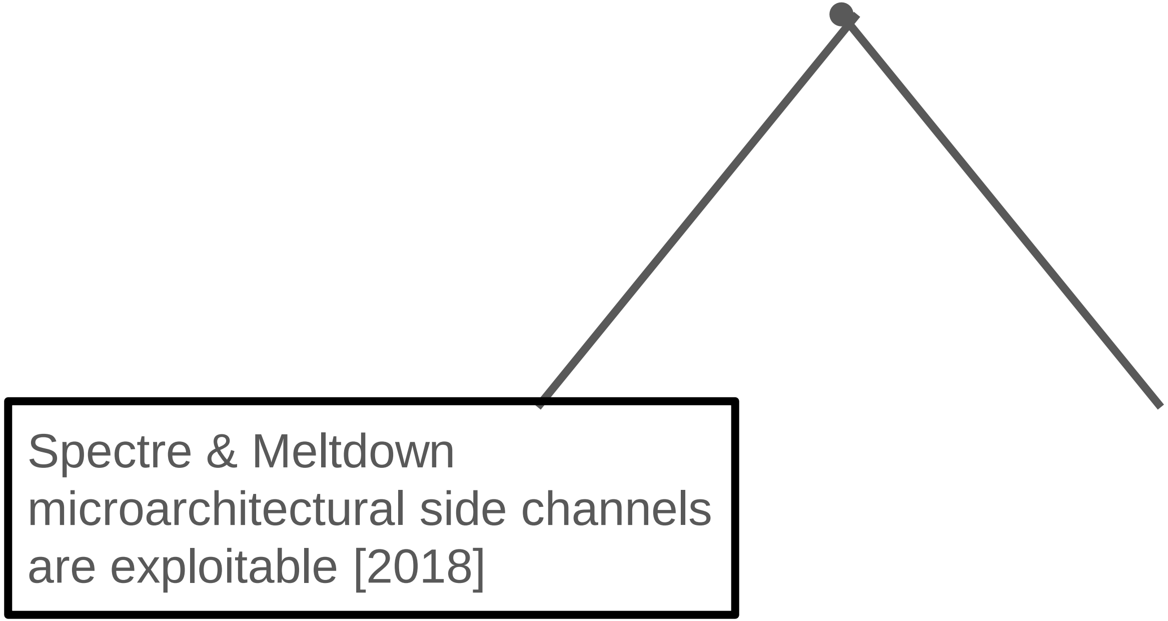
from
Copilot (MS)

Learms context-sent-alciatiion site lifelimes
Conert -senevitly binary veraions, to sample to umbseve calling confeets
Cenoroadatado from unomofived cenofonts
Cancors-alenokivo differer biry vevoons

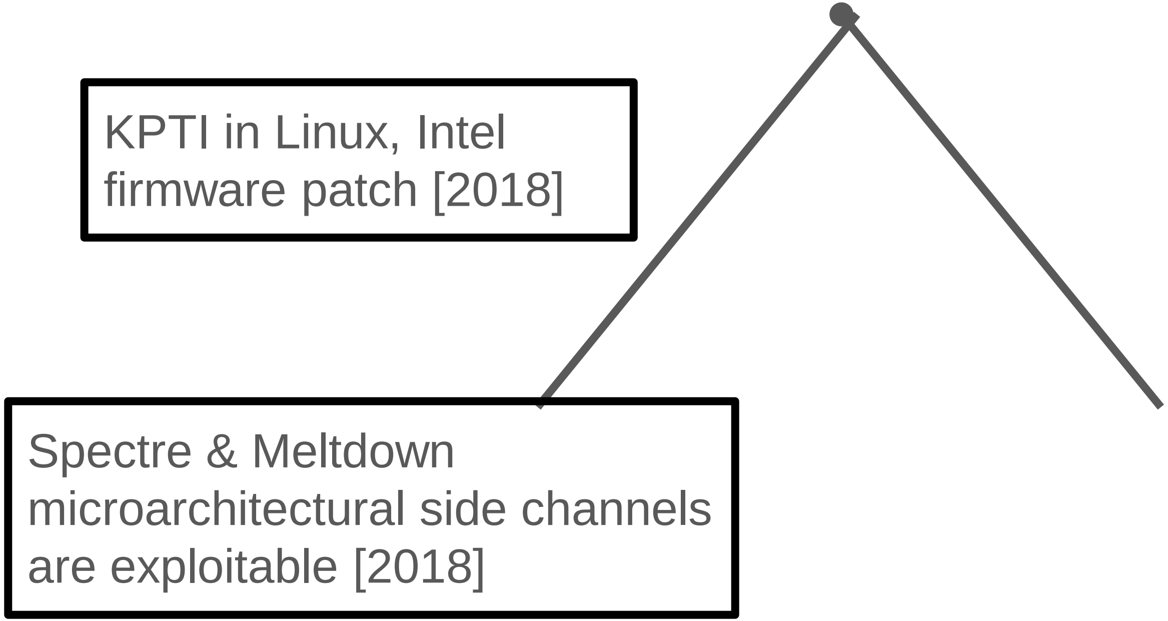
Ap
AS

Hoard [ASPLOS 00] does scalable memory allocation

LLAMA [ASPLOS 20] uses machine learning to reduce fragmentation



Spectre & Meltdown microarchitectural side channels are exploitable [2018]



KPTI in Linux, Intel firmware patch [2018]

Spectre & Meltdown microarchitectural side channels are exploitable [2018]

from
ImageFX (Google)

Spectre & Meltdown
microarchitectural side channels
are exploitable [2018]

from
ImageFX (Google)

Foreshadow [USENIX 18]
Augury [IEEE SP 22]

Spectre & Meltdown
microarchitectural side channels
are exploitable [2018]

from
ImageFX (Google)

Foreshadow [USENIX 18]
Augury [IEEE SP 22]

Spectre & Meltdown
microarchitectural side channels
are exploitable [2018]

????????????????????????????

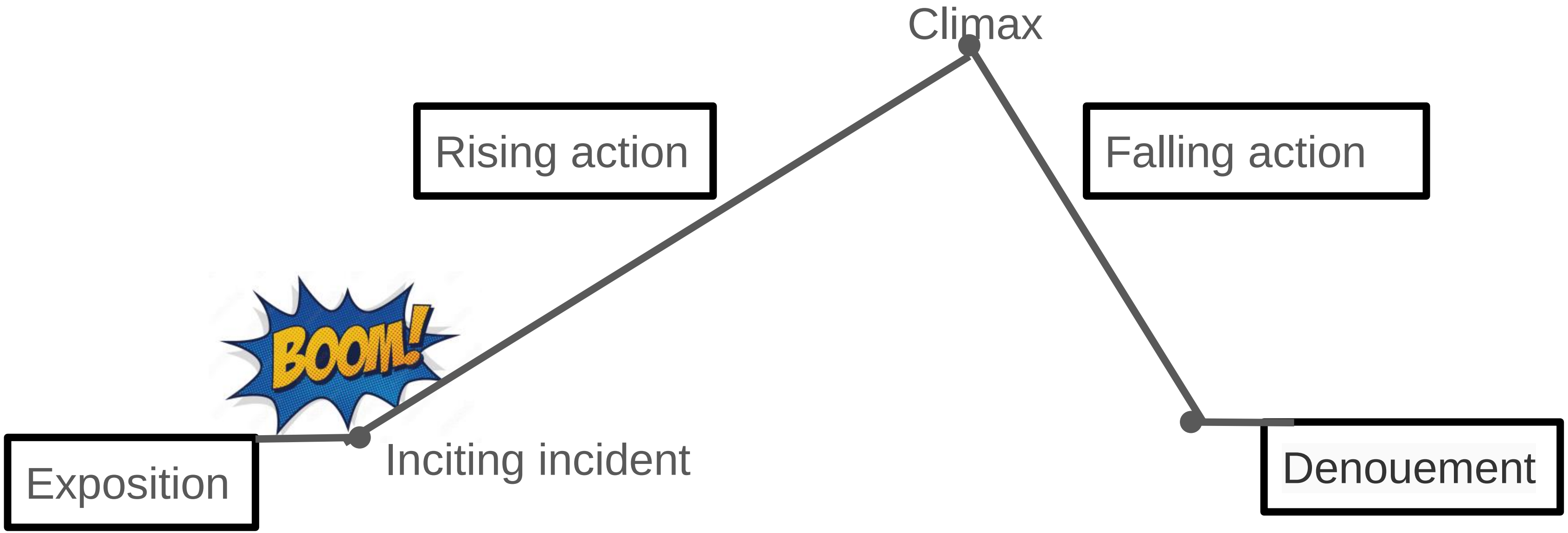
from
ImageFX (Google)

Fo
Au

Spectre & Meltdown microarchitectural side channels are exploitable [2018]

??????????????????????????????

Gustav Freytag’s pyramid [1863]



Climax

Rising action

Falling action

Exposition

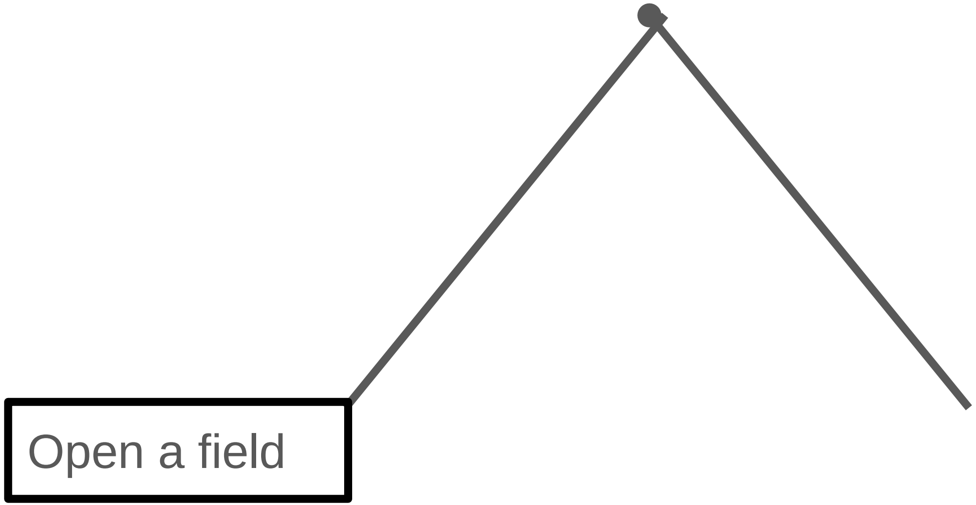
Inciting incident

Denouement

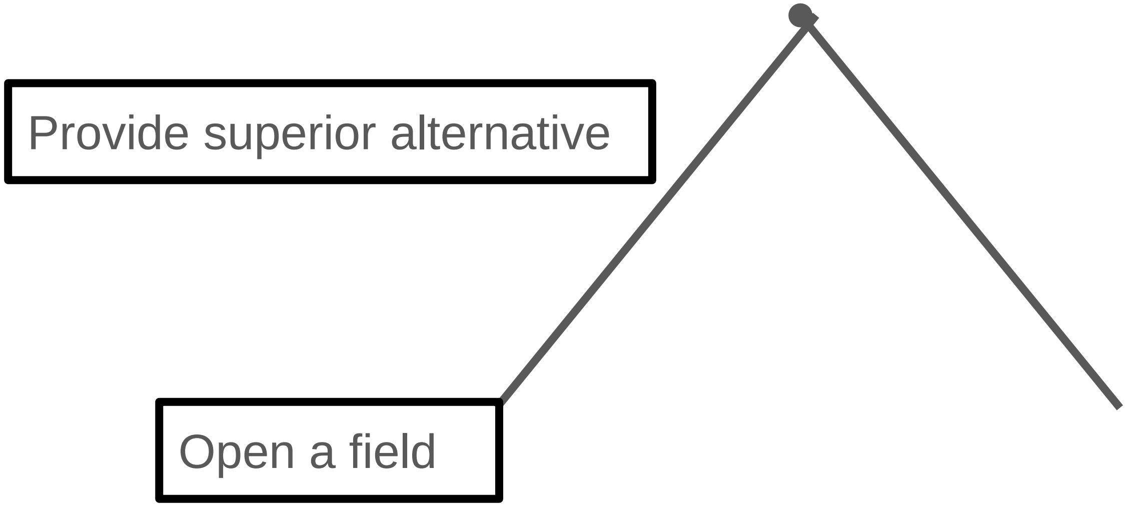
The role of metacognition

- Metadata is data about data, e.g., a file's modification timestamp
  - Metacognition is thinking about thinking
- Metacognition can provide insight and perspective
  - Can get you out of a rut
  - Even if useful, it is no crystal ball

Where is your work in the pyramid?



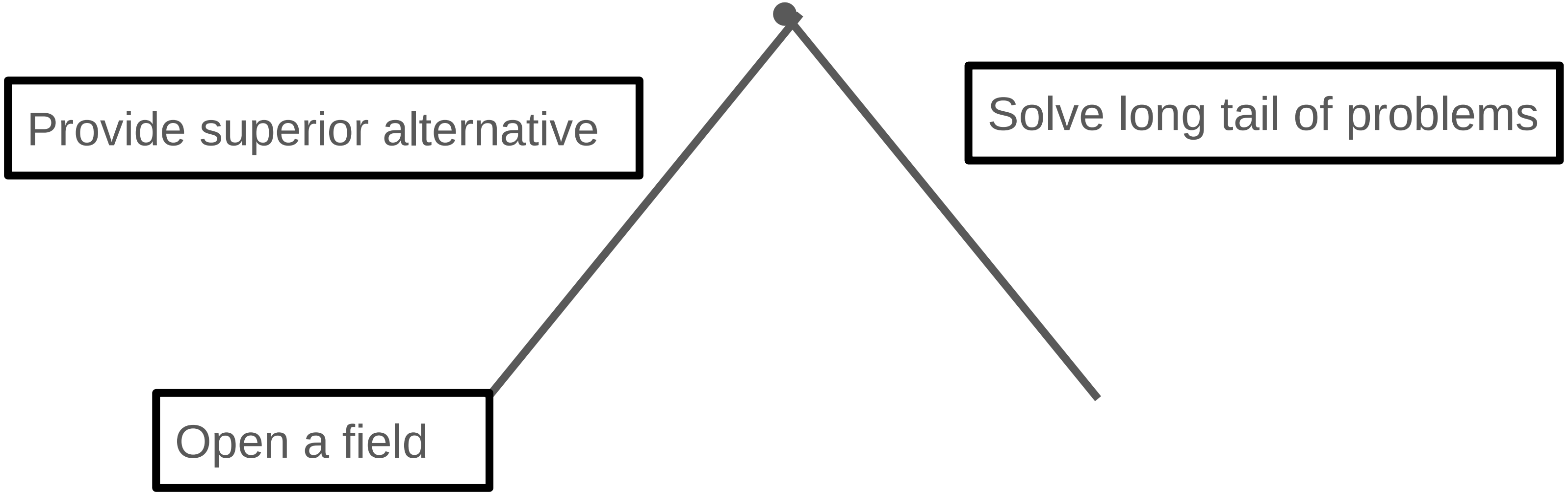
Where is your work in the pyramid?



Provide superior alternative

Open a field

Where is your work in the pyramid?

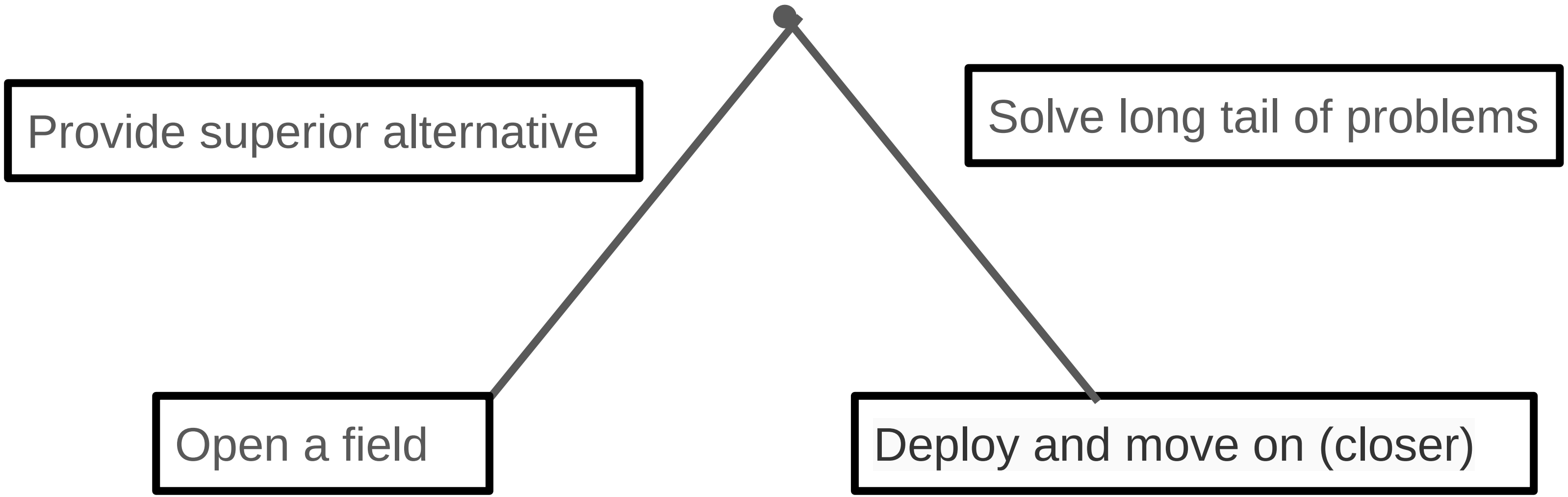


Provide superior alternative

Solve long tail of problems

Open a field

Where is your work in the pyramid?



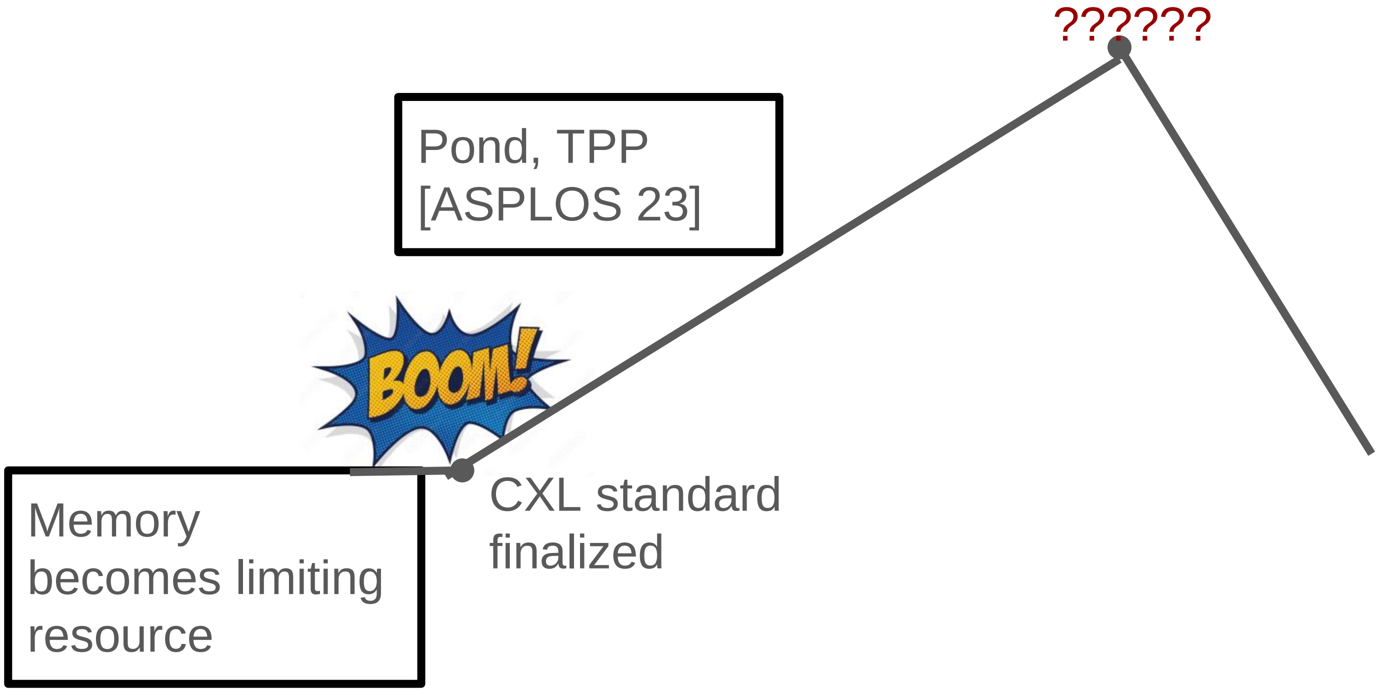
Provide superior alternative

Solve long tail of problems

Open a field

Deploy and move on (closer)

Compute Express Link (CXL) memory



??????

Pond, TPP [ASPLOS 23]

Memory becomes limiting resource

CXL standard finalized

Talk outline

- CXL memory — saving costs
  - Disaggregation motivation
  - CXL memory is transparent
- CXL pods — increasing performance [CXL-SHM SOSP 23]
  - CXL memory is explicitly controlled by programmer
  - Unstructured / global coordination can be fast
- New challenges for CXL pods
  - Tolerating partial failures, why and how

What is a computer?

- [A computer must] store numbers passively—the results of various partial, intermediate calculations. The totality of these organs is called a “memory.”

- John Von Neumann (1958)

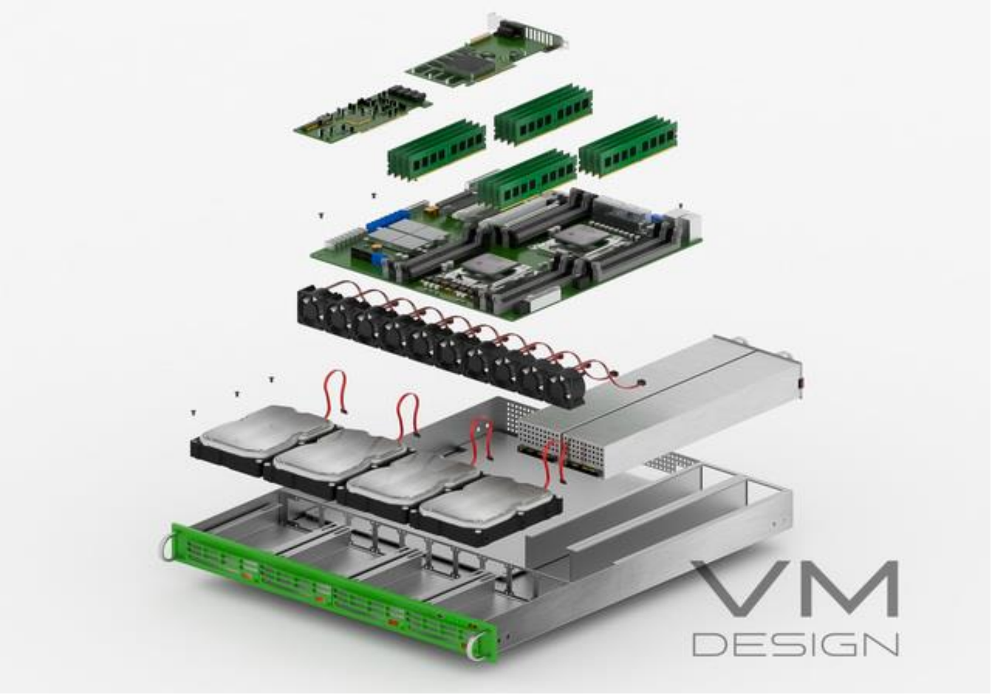


Image credit: cad crowd

What is a computer?

- [A computer must] store numbers passively—the results of various partial, intermediate calculations. The totality of these organs is called a “memory.”

- John Von Neumann (1958)

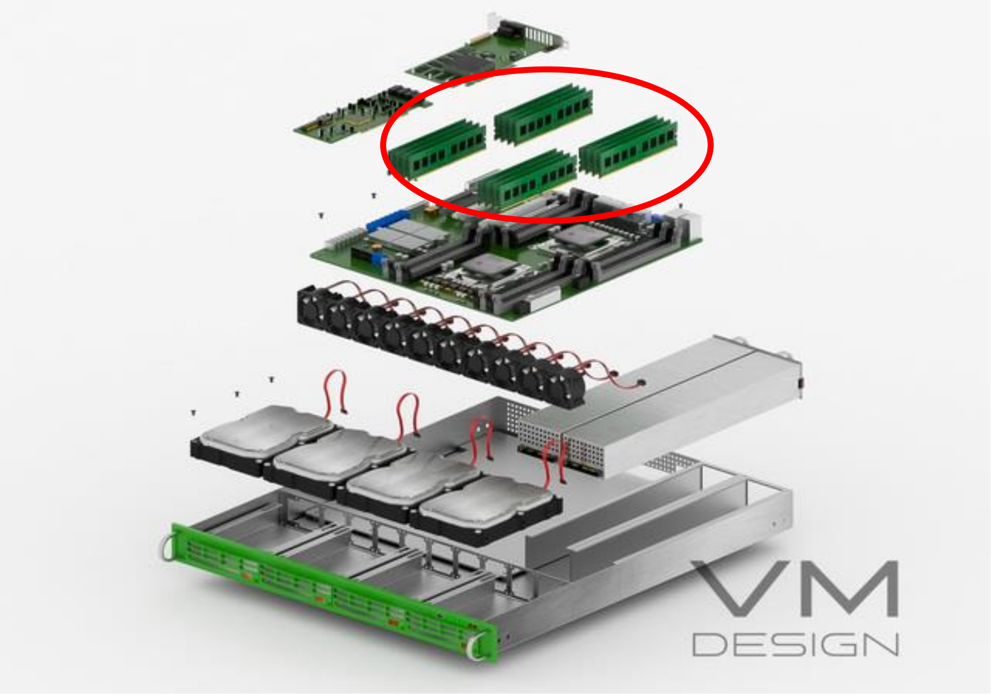
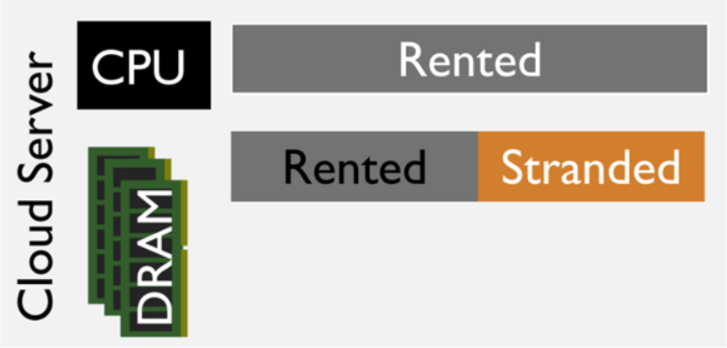


Image credit: cad crowd

Physical racks vs. virtual machines

- Memory stranding in Azure cloud (from Pond [ASPLOS 23])
  - No free CPU cores but memory left
  - Up to 25% stranded memory at 95th percentile
- Untouched memory due to overprovisioning



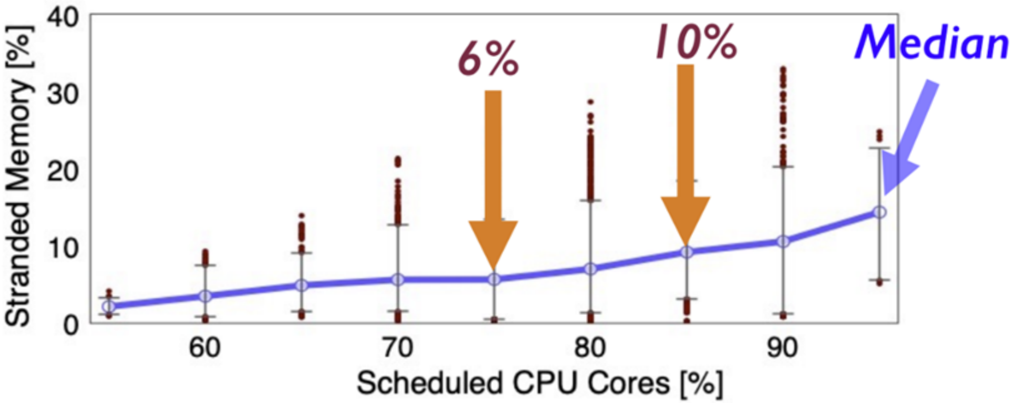
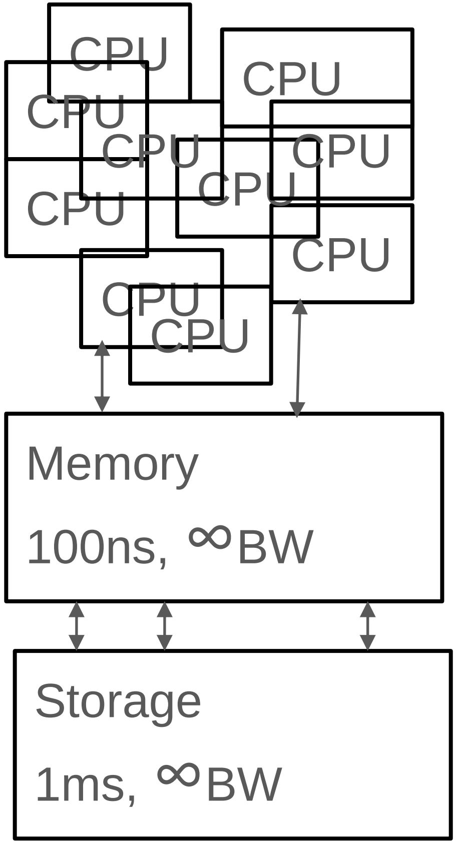


Image credit: Pond [ASPLOS 23]

The dream of disaggregation

- So many compute nodes
- Memory and Storage
  - All the bandwidth you can buy
  - All the capacity you can buy
  - Low latency (physical limits)
- Optimized for cost savings
  - Flexible partitioning of resources
  - Transparent to applications



The reality of disaggregation

A hierarchy of layers

System software!

- Virtual memory was invented for this
- Prediction & migration

Works pretty well

- Pond, TPP, etc.

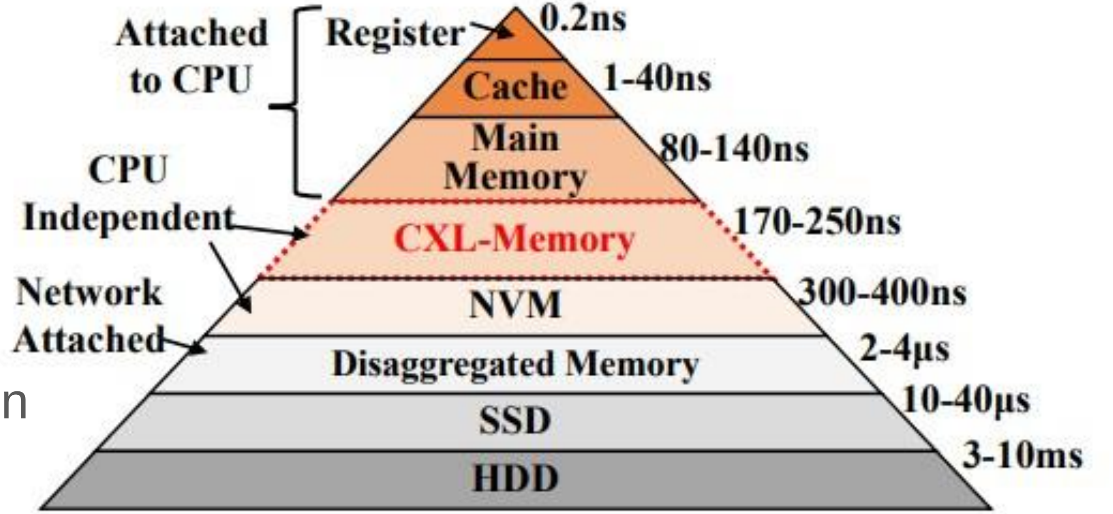


Image credit: Timothy Prickett Morgan, The Next Platform

Questions remain

- Enough layers?
- Enough bandwidth?
- Low enough latency?
- Accurate prediction?
- Latency insensitive applications?
- Active area now

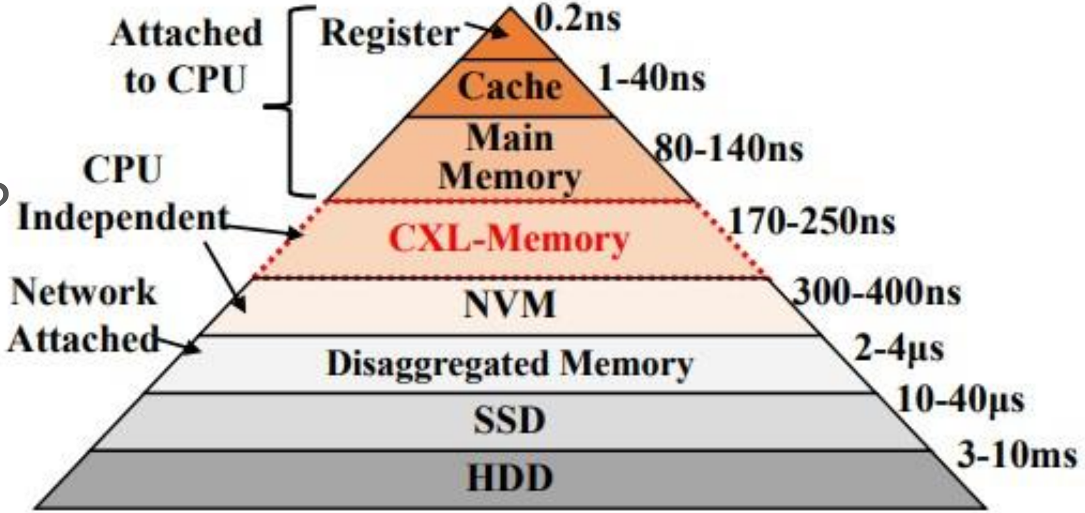


Image credit: Timothy Prickett Morgan, The Next Platform

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

CXL Pod

- Machines connected to CXL memory

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

Single-host software vs. distributed software

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

CXL Pod

- Machines connected to CXL memory

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

A tale of two climates

One Host

- Shared mutable state
- Centralized state
- Many efficient algorithms
- Limited concurrency
- Database

CXL Pod

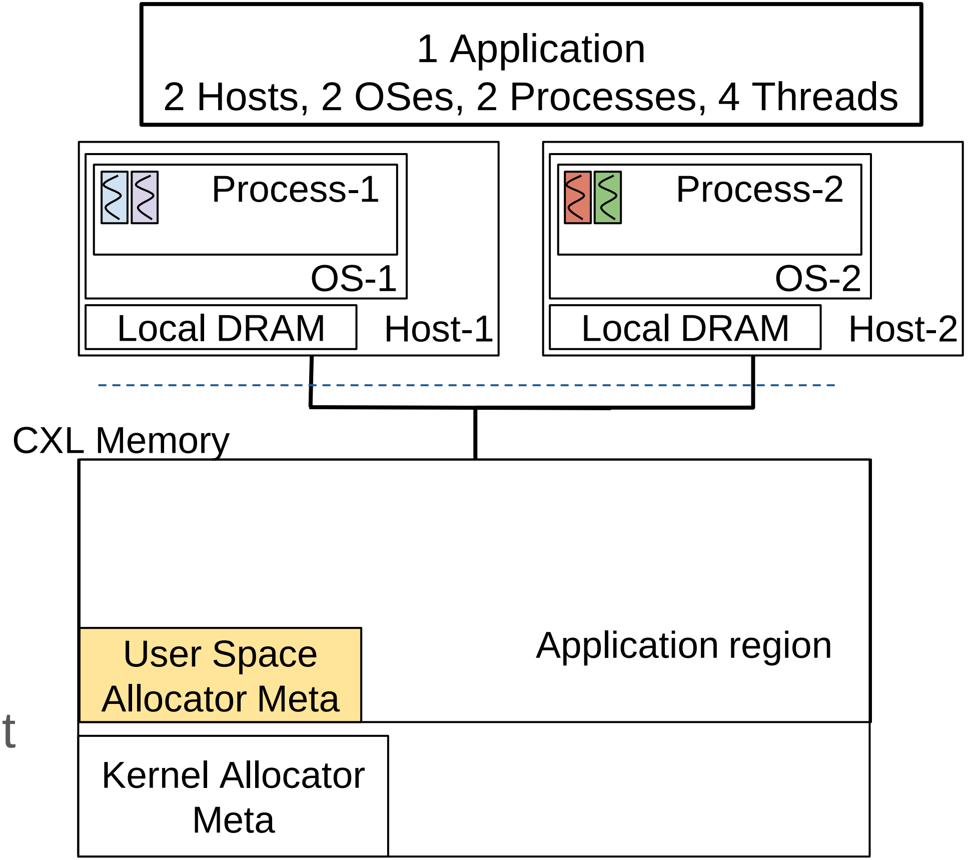
- Encapsulate complexity in data structures
- Low tail latency
- The “SQLite” of distributed systems

Distributed (many hosts)

- Replicated state machines
- Scalable
- Fast failover
- Difficult to construct and maintain (performance)
- Key-value store

CXL Pod

- Runs single-node SW
  - Fine-grain sharing CXL
  - Requires next HW standard
- Need support from
  - OS + memory allocator
- 16 hosts X 288 cores
  - 4,608 cores Sierra Forest
  - 7,200 MapReduce [04]



What will run on a CXL pod?

- An in-memory database
  - High performance
  - High availability, no downtime
  - Coordination by shared memory more efficient than
    - Partitioned state +
    - Distributed transactions over the network
- Long-running computation with lots of state
  - Computation is valuable enough to require fault tolerance
  - Check pointing state is slow
    - Consumes storage bandwidth

What are the requirements for a CXL pod?

- Will I get hardware cache coherence across all CXL?
    - Uncertain at this time
    - Might require SRAM tags
        - Raising the cost
- What should HW provide SW?

Persistent memory: avoid extra instructions

```
MOV X1, 70 ; store 70 to X1
CLWB X1     ; flush X1 from cache
SFENCE
PCOMMIT   ; persist
SFENCE     ; ensure pcommit finished
```

- Before 2016, pcommit needed

Persistent memory: avoid extra instructions

```
MOV X1, 70 ; store 70 to X1
CLWB X1     ; flush X1 from cache
SFENCE
```

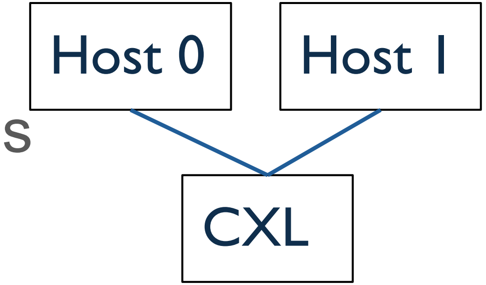
- After 2016
    - pcommit deprecated

Persistent memory: avoid extra instructions

```
MOV X1, 70 ; store 70 to X1
SFENCE
```

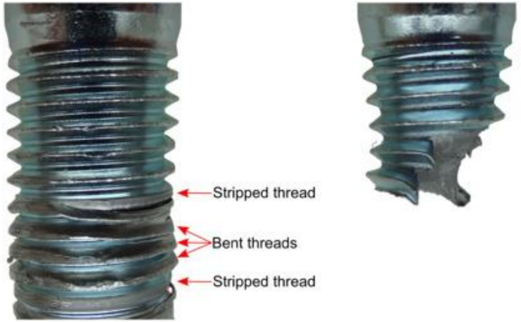
- After 2020
  - Extended asynchronous DRAM Refresh (eADR)
  - No more cache flushing
- Analogy for CXL: global persistent flush (GPF)
  - No more performance sapping clwb!

Challenges of the CXL pod - partial failure



- Let’s say one OS reboots or one process dies
  - Do I have to restart all OSes (or all processes)?
  - Full restart is bad for availability
- Tolerating partial failure means
  - Application remains available during partial recovery
  - OS / process recovers and rejoins
- CXL pod fault model
  - Is it a shared memory multiprocessor or a distributed system?
  - Distributed systems should tolerate partial failures

What goes wrong on a partial failure?

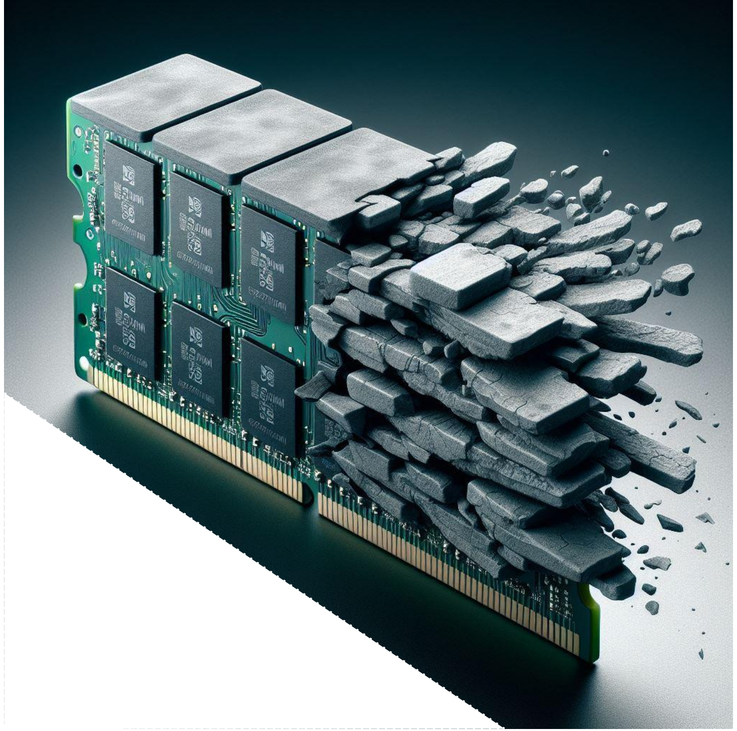


a) Thread failure (PT bolt, L_t = 9 mm) b) Bolt fracture (PT bolt, L_t = 11 mm)

- Shared data structures go in shared CXL
  - Shared data structures need synchronization
- OSes & applications have to synchronize on CXL memory
  - Spinlocks, futexes, mutexes, semaphores are not fault-tolerant
  - Die with a lock held ⇒ Deadlock
- OS reboot is not a global quiescent point!
  - Can’t rebuild DRAM from PM on OS reboot [NOVA FAST 16]
- On recovery, restore state from where? Storage is slow

CXL pod partial failure model

- Make CXL memory persistent
  - Give it independent power supply
  - Protect integrity with ECC
    - Raising cost of module
- On a partial failure restore from CXL memory state
  - Applications remain available during recovery
- How do we synchronize and remain fault-tolerant?



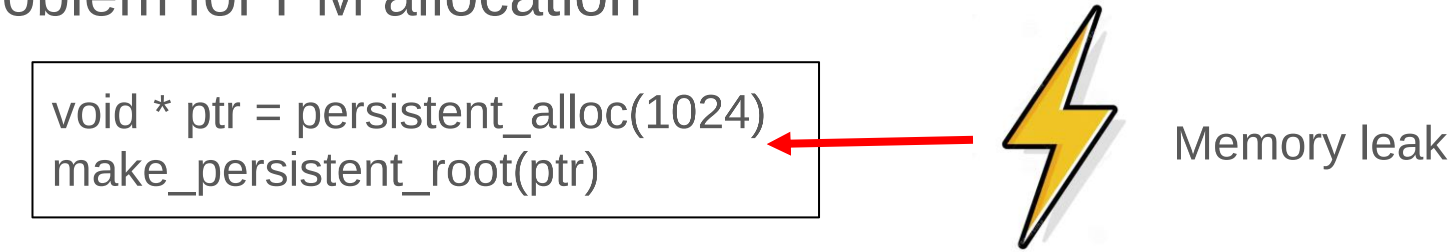
Transactions to the rescue!

- Transactions are fault-tolerant
  - Persistent memory systems use them for memory allocation
- Problem for PM allocation

```
void * ptr = persistent_alloc(1024)
make_persistent_root(ptr)
```

Transactions to the rescue!

- Transactions are fault-tolerant
  - Persistent memory systems use them for memory allocation
- Problem for PM allocation



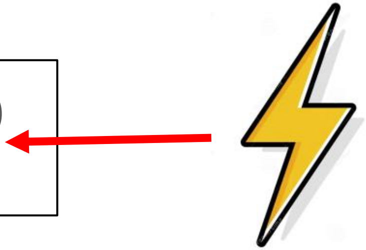
```
void * ptr = persistent_alloc(1024)
make_persistent_root(ptr)
```

Memory leak

Transactions to the rescue!

- Transactions are fault-tolerant
  - Persistent memory systems use them for memory allocation
- Problem for PM allocation

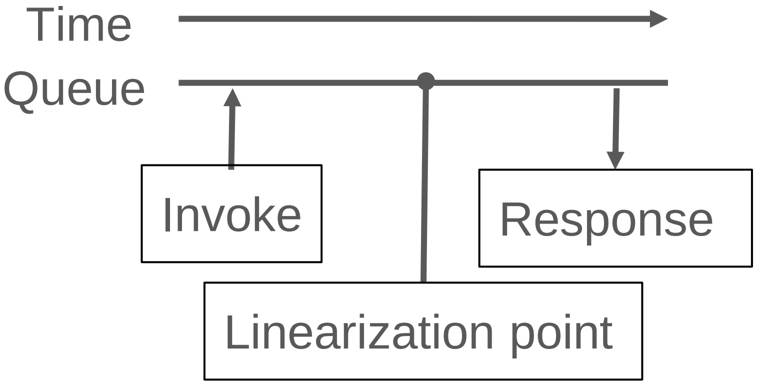
```
void * ptr = persistent_alloc(1024)
make_persistent_root(ptr)
```



Memory leak

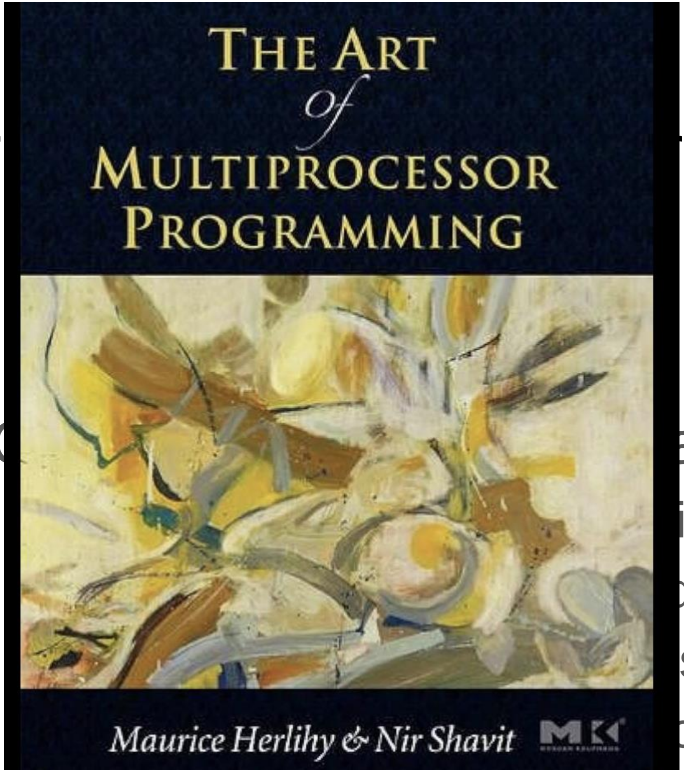
- Tolerating partial failures more pervasive than memory allocation
  - Let’s avoid mandating fully transactional programming model
  - Find efficient special-case solutions

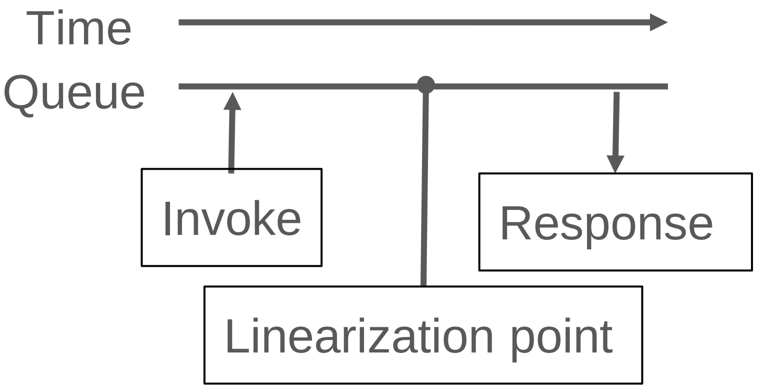
Correctness under concurrency



- Concurrent safety by linearizability [Herlihy, Wing 1990]
    - Operations have linearization point between invocation & response
        - Respects real-time order
    - Reorder linearization points to be sequential
    - Sequential history is correct for sequential specification of object
- But linearizability says nothing about failures
    - Use durable linearizability [Izraelevitz 2016]

Cor ency


THE ART of MULTIPROCESSOR PROGRAMMING
Maurice Herlihy & Nir Shavit
MK



- C arizability [Herlihy, Wing 1990]
    - ion point between invocation & response
    - der
    - s to be sequential
    - ct for sequential specification of object
- But linearizability says nothing about failures
    - Use durable linearizability [Izraelevitz 2016]

Correctness under concurrency + partial failure

- Durable linearizability has limitations for partial failure
- Need detectable execution [Friedman 2018]
    - Need the ability to execute operations exactly once
        - Crash while enqueue object O into Q
        - On recovery did I enqueue?
        - Can look for O in Q, but another thread might have dequeued it
    - Recovery settles question of whether operation succeeded
- Need linearizability + detectable execution

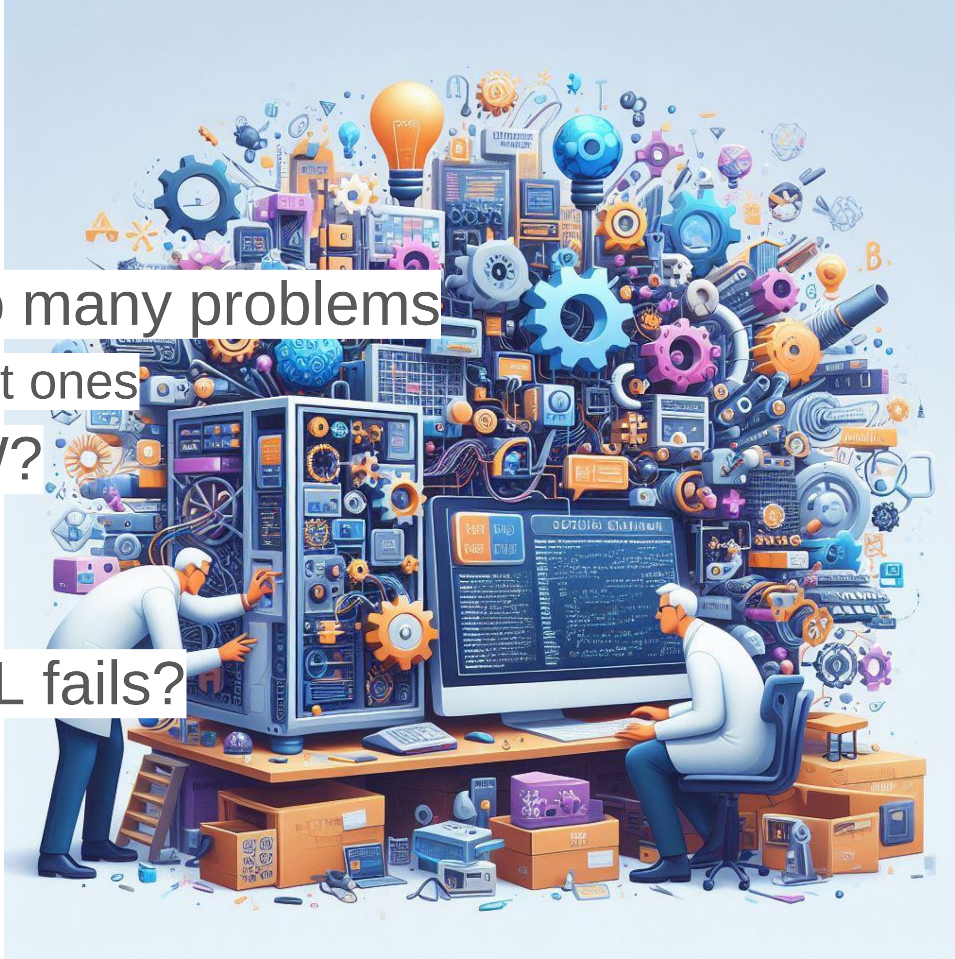
Performance of OpenMPI broadcast microbenchmark

- OSU microbenchmark across 16 VMs
- Message passing / distributed system benchmark
- Memory is more efficient than network messages

	OpenMPI (µs)		**CXL (µs)**	
Size	**p50**	**p99**	**p50**	**p99**
64B	18.5	53.7	7.2 (2.6x)	12.9 (4.2x)
1MB	3120	3660	406 (7.7x)	439 (8.3x)

Promise for CXL and beyond

- Mathematically, there are too many problems
  - Technology identifies important ones
- What should HW provide SW?
  - Vital as HW stops scaling
  - Ease SW programming model
- What do we learn even if CXL fails?
  - Break down solutions
  - Use the parts in new systems



Why do we do research?

Why do we do research?

- Ego gratification

Why do we do research?

- Ego gratification
- Impact

Why do we do research?

- Ego gratification
- Impact
    - Change the world

Why do we do research?

- Ego gratification
- Impact
  - Change the world
  - Positive effect on other people and society

Why do we do research?

- Ego gratification
- Impact
  - Change the world
  - Positive effect on other people and society

Impact

Why do we do research?

- Ego gratification
- Impact
  - Change the world
  - Positive effect on other people and society

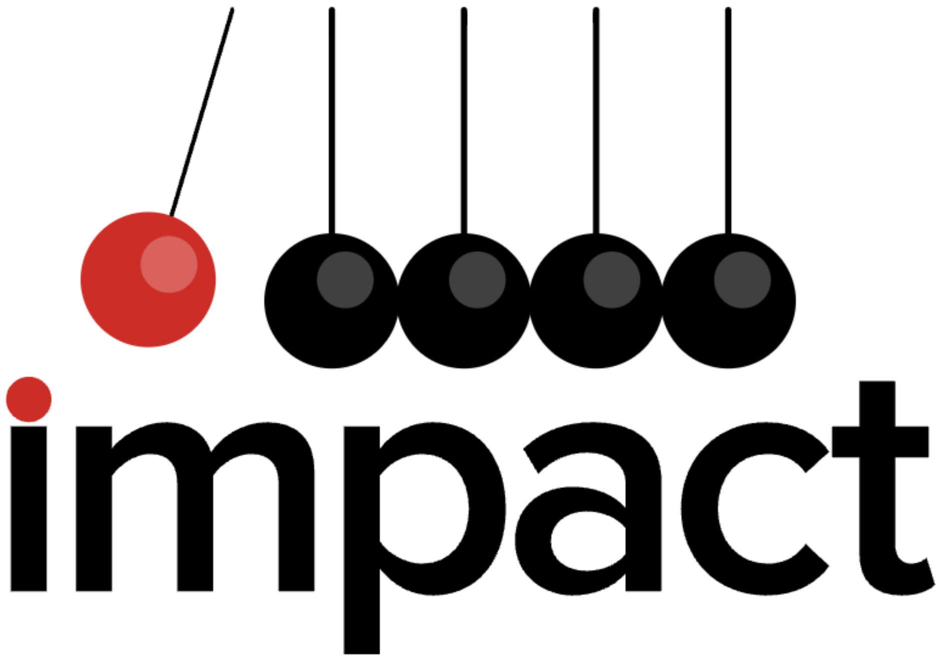
Impact

Impact

Impact

Impact

Why do we do research?



Why do we do research?

Image Source: liorpt/123RF

Why do we do

Image Source: liorpt/123RF

Why do we do research?

- Ego gratification
- Impact
  - Change the world
  - Positive effect on other people and society
- Aha moment, pursuit of truth
  - Ph.D: Academic degree that pushes boundaries of human knowledge in a specialized field through focused research for several years
  - Insight is hard to search for and hard to recognize

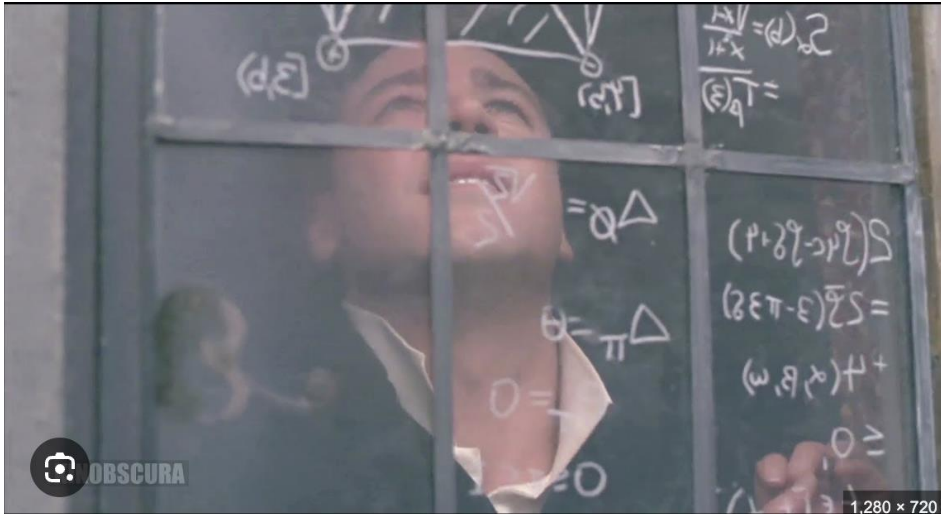
What is the nature of insight?

Credit: Good Will Hunting, A Beautiful Mind, Its Always Sunny in Philadelphia

What is the nature of insight?

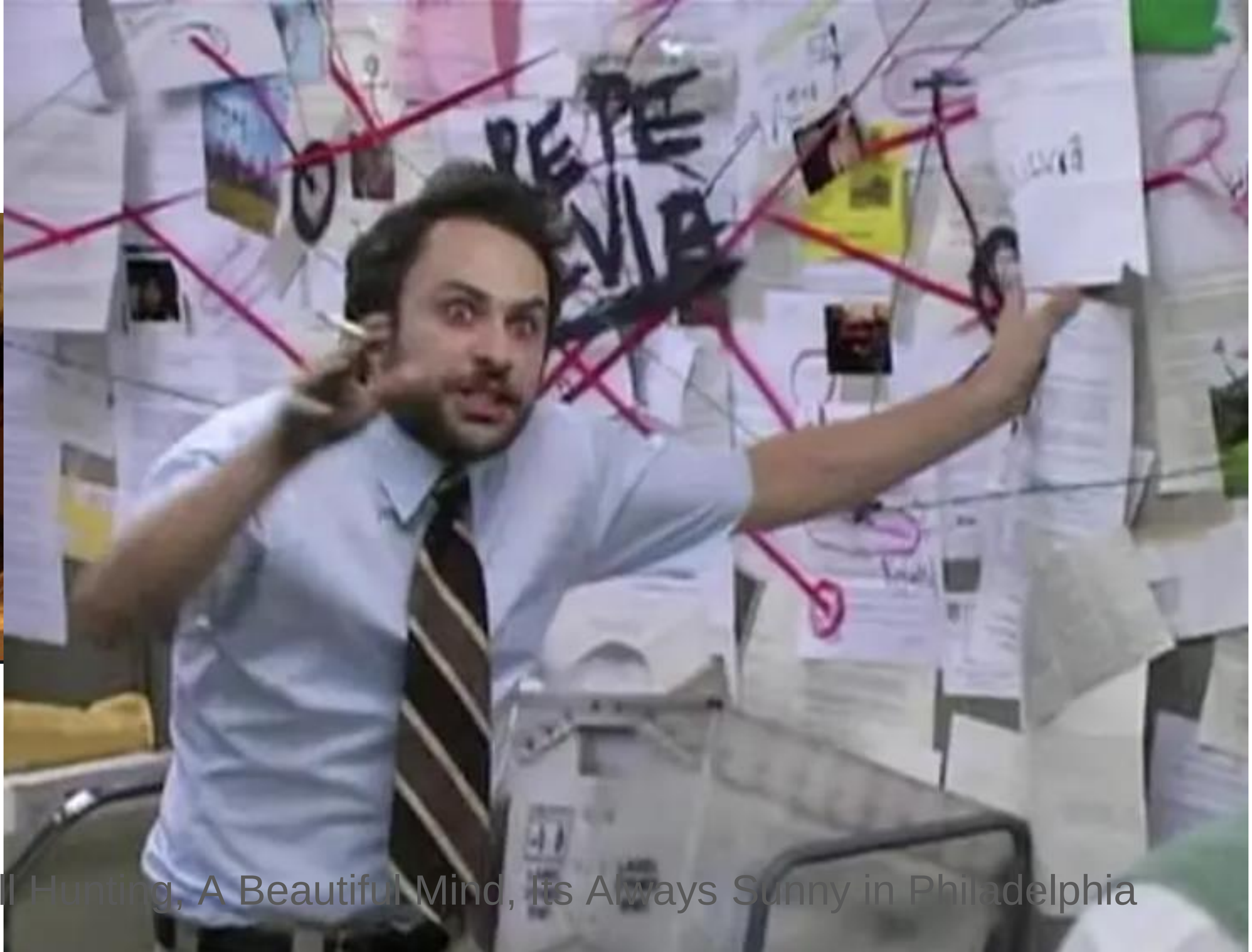
Credit: Good Will Hunting, A Beautiful Mind, Its Always Sunny in Philadelphia

What is the nature of insight?



Credit: Good Will Hunting, A Beautiful Mind, Its Always Sunny in Philadelphia

What is the



Credit: Good Will Hunting, A Beautiful Mind, Its Always Sunny in Philadelphia

Insight arises within a group

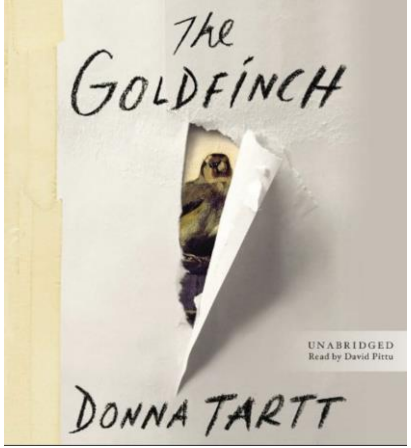
- Research is a social activity
    - A research group
    - The research community
- The whole is greater than the sum of its parts
    - I write papers because I learn so much from writing them
    - My old papers are written by someone more knowledgeable than I
- Unreasonable levels of effort help
    - Dedication displaces normality
    - Synesthesia

Research for the long haul

- Study what you love and what you are good at
- Explore, but topics recur in popularity
- Find the right fit

Research for the long haul

- Study what you love and what you are good at
- Explore, but topics recur in popularity
- Find the right fit



> Every shrink, every career counselor, every Disney princess knows the answer: “Be yourself.” “Follow your heart.”
>
> Only here's what I really, really want someone to explain to me. What if one happens to be possessed of a heart that can't be trusted?
> --Donna Tartt, The Goldfinch

How do we remain a robust community?

- Number of submissions is way up
- Number of accepted papers is way up
- Size of program committees is way up
- What do we do?
    - One or two annual deadlines, not three
    - History of paper reviews from previous conferences?
    - Pay per submission (in cash, in reviews)

Research ethics

No matter what our place in life is, each human being possesses a fundamental inner freedom that cannot be compromised unless we let it. And that therefore imbues us with an innate demand for personal responsibility.

- Like Stories of Old
  - https://youtu.be/FDVR73qUSXU?si=Bd-bUzOuZ4-BXepE&t=1307

Many thanks

Zhiting Zhu
UT Austin

Newton Ni
UT Austin

Nam Sung Kim
UIUC

Zhipeng Jia
Google

Yibo Huang
UT Austin

Yan Sun
UIUC

Summary

- CXL memory — saving costs
  - Disaggregation motivation
  - CXL memory is transparent
- CXL pods — increasing performance
  - CXL memory is explicitly controlled by programmer
  - Unstructured / global coordination can be fast
- New challenges for CXL pods
  - Tolerating partial failures

Summary

- CXL memory — sav
  - Disaggregation mot
  - CXL memory is tran
- CXL pods — increa
  - CXL memory is exp
  - Unstructured / globa
- New challenges for
  - Tolerating partial fa