How to do the paper/ talk reviews?

1. Summary
   * What is the problem the paper is trying to solve?
   * What are the key ideas of the paper? Key insights?
   * What is the key contribution to literature at the same time it was written?
   * What are the most important things you take out of it?
2. Strengths
   * Does the paper solve the problem well?
3. Weaknesses
   * This is where you should think critically. Every paper/idea has a weakness. This does not mean the paper is necessarily bad. It means there is room for improvement and future research can accomplish this.
4. Can you do much better? Present your thoughts/ideas.
5. What have you learned/enjoyed/disliked in the paper? Why?

Review should be short and concise (~half a page or shorter)

Advice on paper/talk reviews

* When doing the reviews, be very critical
* Always think about better ways of solving the problem or related problems
* Do background reading
  + Reviewing a paper/talk is the best way of learning about a research problem/topic
* Think about forming a literature survey topic or a research proposal

What I googled for and read about them, which I was not familiar with in the slide.

**Expressive Memory**: There is a paper and also a presentation on YouTube about this. I watched that presentation and noted what I learned about “expressive memory” below.

Only hardware (like caches, prefetchers, memory controllers) and software (imposing programmability and portability challenges) optimizations independently is not enough to get higher performance for new applications like Machine Learning. Also, with growing HW/SW sophistication, traditional interfaces limit optimization effectiveness, so it is time for a richer interface between hardware and software. “XMem” or expressive memory is proposed by **prof. Onur Mutlu** research team to solve this challenge. Expressive memory is a new cross-layer abstraction which interferes between the application and the underlying hardware. Expressive memory communicates high level program semantics from the application to the underlying system and architecture which can then be used to aid in performance optimization. Expressive memory provides a new way to perform many traditional hardware and software optimizations (like: Cache Management, Data placement in DRAM, Data Compression, Approximation, DRAM cache Management, NVM management, NUCA/ NUMA Optimization, etc). Exposing key program information allows the hardware to better adapt to the application and thereby provide better program ability, portability and resource efficiency.

**I watched Future Computing Architecture Presentation on YouTube and I noted down what I learned.**

Prof. Onur Mutlu is from Kapadoklya, Turkey.

**Why do we do computing?** To solve problems (Prof.’s answer)| to gain insight – not just number crunching (Hamming’s answer) - 1962| to enable better life and future

**How does a computer solve problems?** By orchestrating electrons

**How do problems get solved by electrons?** With the help of the transformation hierarchy (the broader view of computer architecture)

**What is computer architecture?** It is the **science/engineering** and **art** of designing computing platforms (hardware, interface, system SW, and programming model) to achieve a set of **design goals**. Different platforms demand different goals.

**Our axiom (assumption or maybe presumption) as computer architects**: To achieve highest energy efficiency and performance we must take the expanded view of computer architecture and co-design across the hierarchy from algorithms to devices and specialize as much as possible within the design goals.

**What kind of a future do we want?** We as designers design for future not for now. In first step we want a **reliable**, **secure**, and **safe** future. Secondly, we want a **sustainable** and **energy-efficient** one (better quality of life). Then we need a **high performance** to solve the toughest and all problems. Also, the future is more **personalized** and **private** in every aspect of life: health, medicine, spaces, devices, robotics, etc.

**Increasingly demanding applications**: Dream and applications will come. As applications push boundaries, computing platforms will become increasingly strained.

**Three key issues in future platforms**:

1. Fundamentally secure/reliable/safe architectures
2. Fundamentally energy-efficient architectures
   * Memory-centric (data-centric) architectures
3. Architectures for Genomics, Medicine, Health

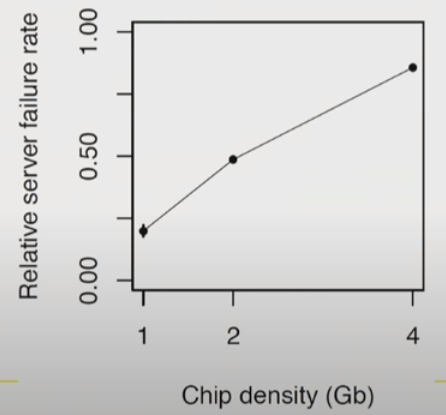
**1 Secure/ Reliable/ Safe Architecture**

**Security**: it is about preventing unforeseen consequences.

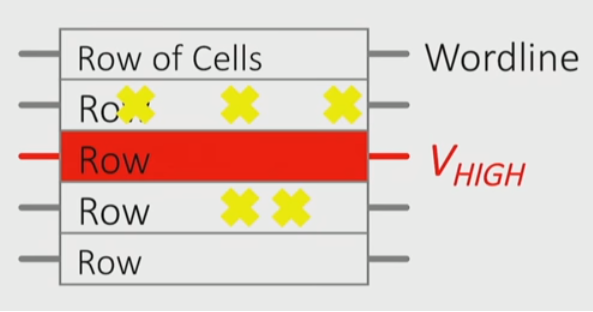
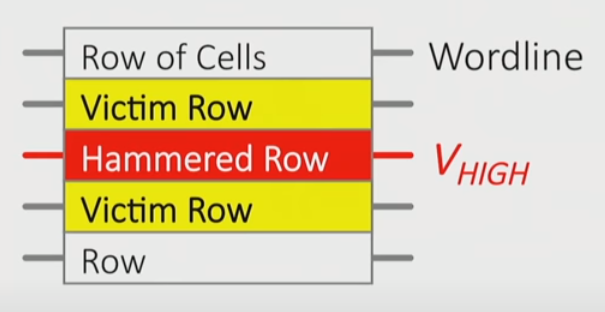
**One major problem**: We do not seem to have design principles for guaranteeing reliability and security.

**Focus is on**: data storage systems (memory) – memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor … Main memory must scale (in size, technology, efficiency, and management algorithms) to maintain performance growth and technology scaling benefits.

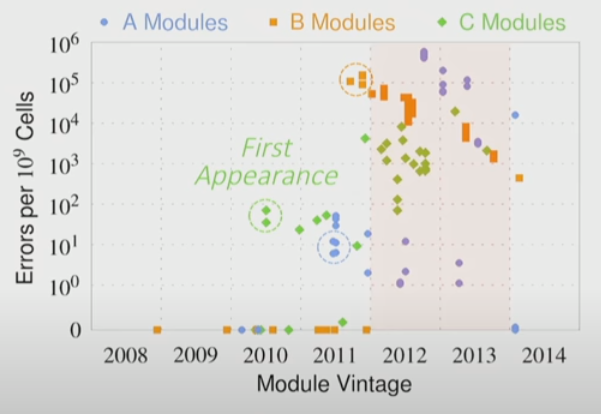
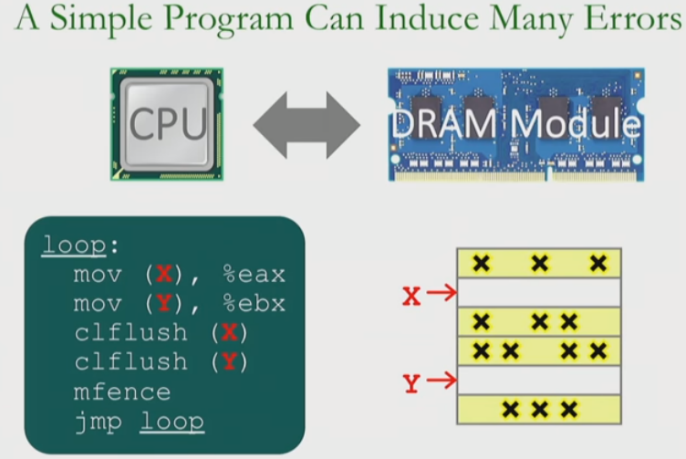
**As Memory scales, it becomes unreliable, data from all of Facebook’s servers worldwide proves this.**



**DRAM RowHammer**: A simple hardware failure mechanism can create a widespread system security vulnerability.

More than 80% of the DRAM chips tested by Prof.’s group were vulnerable to these errors. This is a scaling problem because this does not happen before 2010 but in 2012 and 2013 all the chips that were manufactured exhibit these errors and doesn’t depend on the manufacturer.

Google researchers **Mark Seaborne** and **Thomas Dooley** actually showed that you could gain kernel privileges by exploiting this little hardware vulnerability.

**RowHammer generally**: It is like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after.

**Apple’s patch for RowHammer**: This issue was mitigated by increasing memory refresh rates.

How do we keep memory secure?

1. **Understand**: Methodologies for failure modeling and discovery (modeling and prediction based on read device data)
2. **Architect**: principled co-architecting of system and memory (good partitioning of duties across the stack)
3. **Design and Test**: Principled design, automation, testing (High coverage and good interaction with system reliability methods)

**There are two other solutions**

1. **New technologies**: replace and augment DRAM with a different technology like **NVM**
2. **Embracing un-reliability**: Design memories with different reliability and store data intelligently across them

**Fundamental Solutions to security require co-design across the hierarchy**.

**2 Energy-Efficient Architectures (Memory-centric (data-centric) architectures)**

Today, data access is the major performance and energy bottleneck, and our current design principles cause great energy waste. Processing of data is performed far away from the data at great system cost.

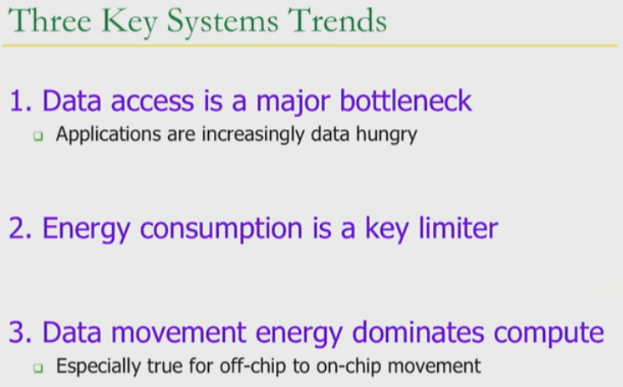
**Perils of processor-centric design**

1. **Grossly imbalanced systems**

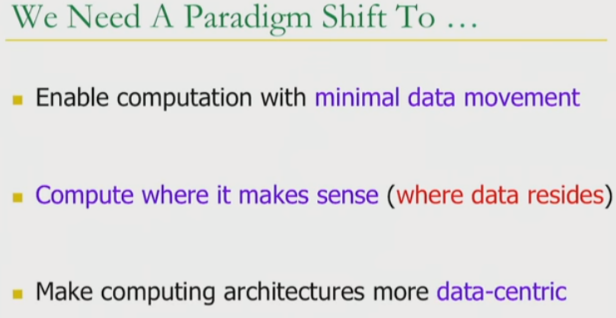
* Processing done only in one place
* Everything else just stores and moves data: data moves a lot
  + Energy Inefficient
  + Low Performance
  + Complex

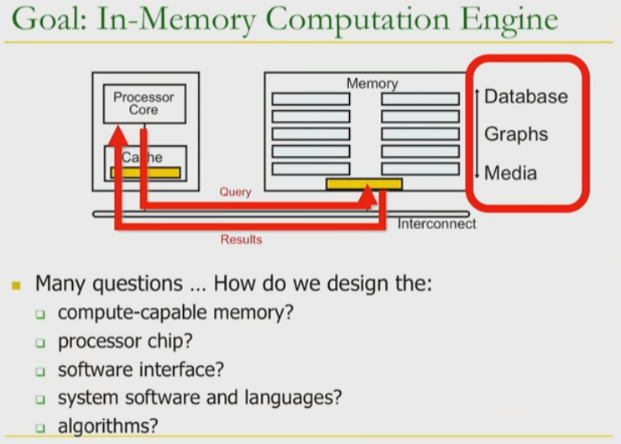
1. **Overly complex and bloated processor (and accelerators)**

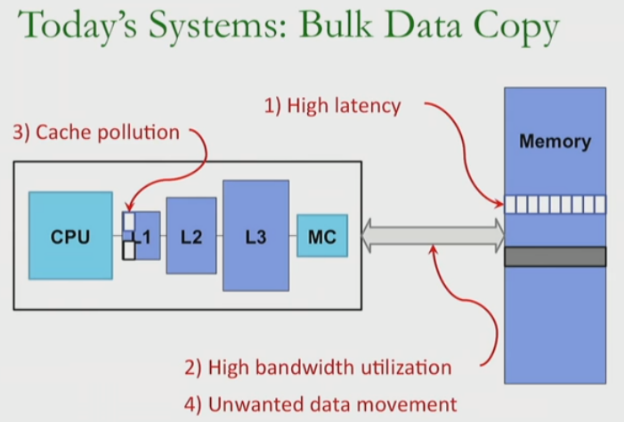
* To tolerate data access from memory
* Complex hierarchies and mechanism
  + Energy Inefficient
  + Low performance
  + Complex

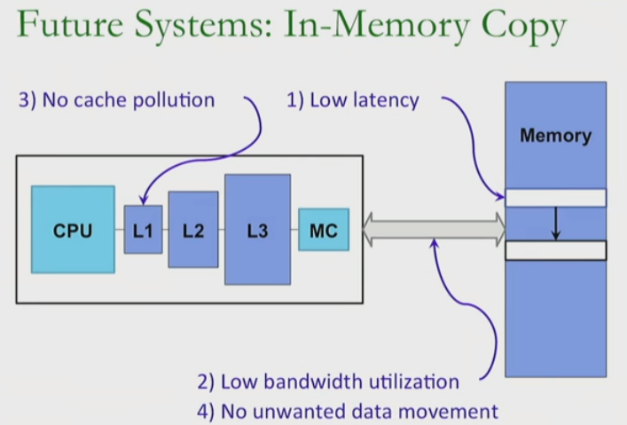


**Bitter Fact**: A memory access consumes ~1000X the energy of a complex addition.









**3 Architectures for Genomics, Medicine, Health**

**Genome** Analysis: **Sequencing** – **mapping** – **variant calls** – **scientific discovery**

Bottleneck is mapping, we can sequence very fast.