Computer Architecture
Lecture 6a: ChargeCache

Hasan Ibrahim Hasan
ETH Zürich
Fall 2018
4 October 2018

ChargeCache
Reducing DRAM Latency by Exploiting Row Access Locality

Hasan Hassan,
Gennady Pekhimenko,
Nandita Vijaykumar,
Vivek Seshadri, Donghyuk Lee,
Oguz Ergin, Onur Mutlu

SAFARI
Carnegie Mellon
TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY
kasirga

Executive Summary

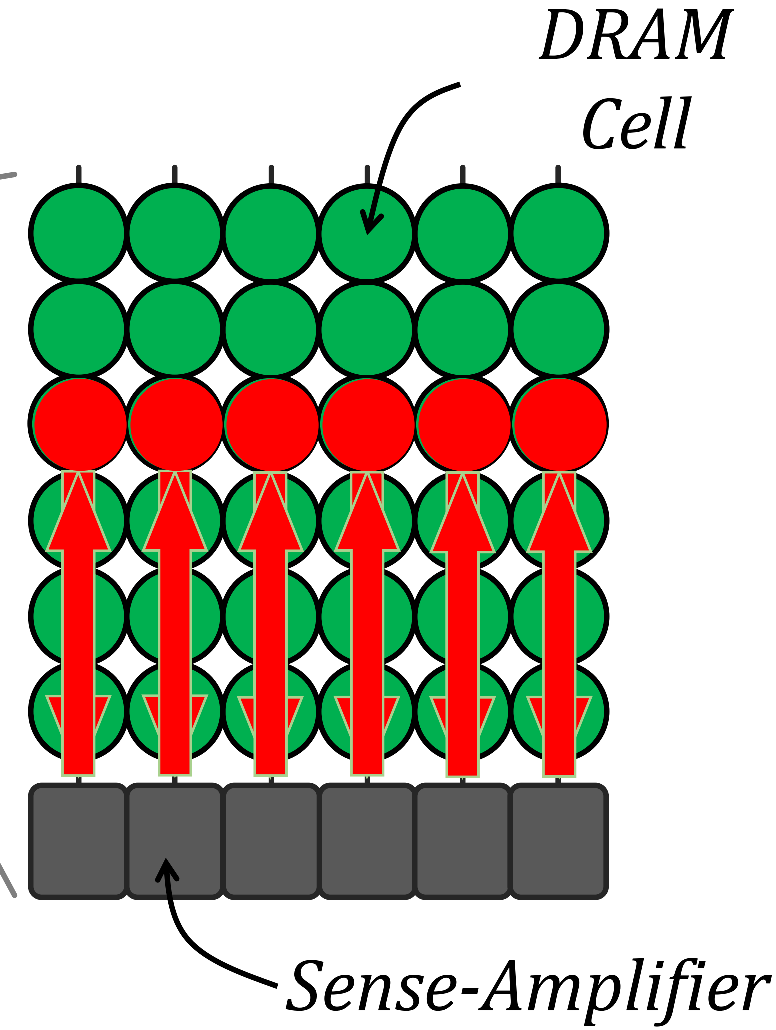
- **Goal**: Reduce average DRAM access latency with no modification to the existing DRAM chips
- **Observations**:
  1) A highly-charged DRAM row can be accessed with low latency
  2) A row's charge is restored when the row is accessed
  3) A recently-accessed row is likely to be accessed again**:**

     Row Level Temporal Locality (RLTL)
- **Key Idea**: Track recently-accessed DRAM rows and use lower timing parameters if such rows are accessed again
- **ChargeCache:**
  – Low cost & no modifications to the DRAM
  – Higher performance (**8.6-10.6%** on average for 8-core)
  – Lower DRAM energy (**7.9%** on average)

Outline

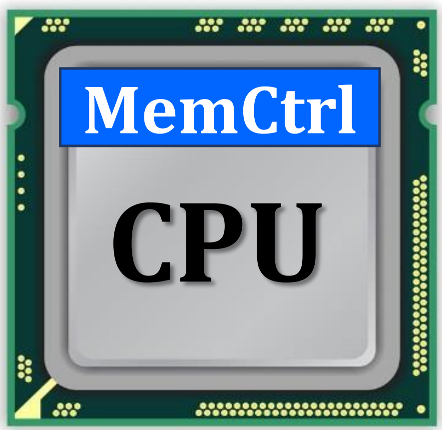
1. **DRAM Operation Basics**
2. Accessing Highly-charged Rows
3. Row Level Temporal Locality (RLTL)
4. ChargeCache
5. Evaluation
6. Conclusion

DRAM Stores Data as Charge

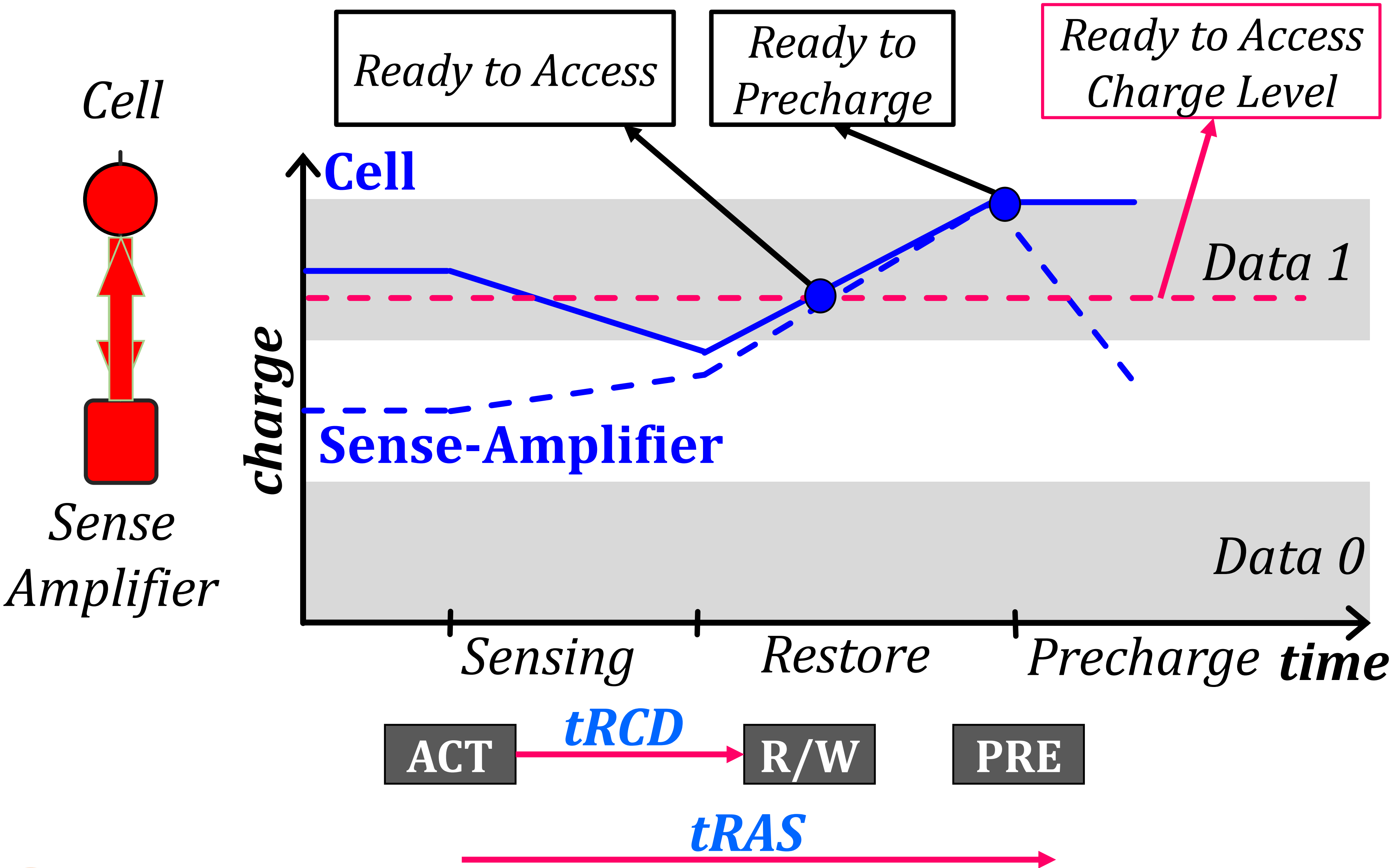


Three steps of charge move

1. S
2. R
3. Precharge



DRAM Charge over Time



Outline

1. DRAM Operation Basics

2. Accessing Highly-charged Rows

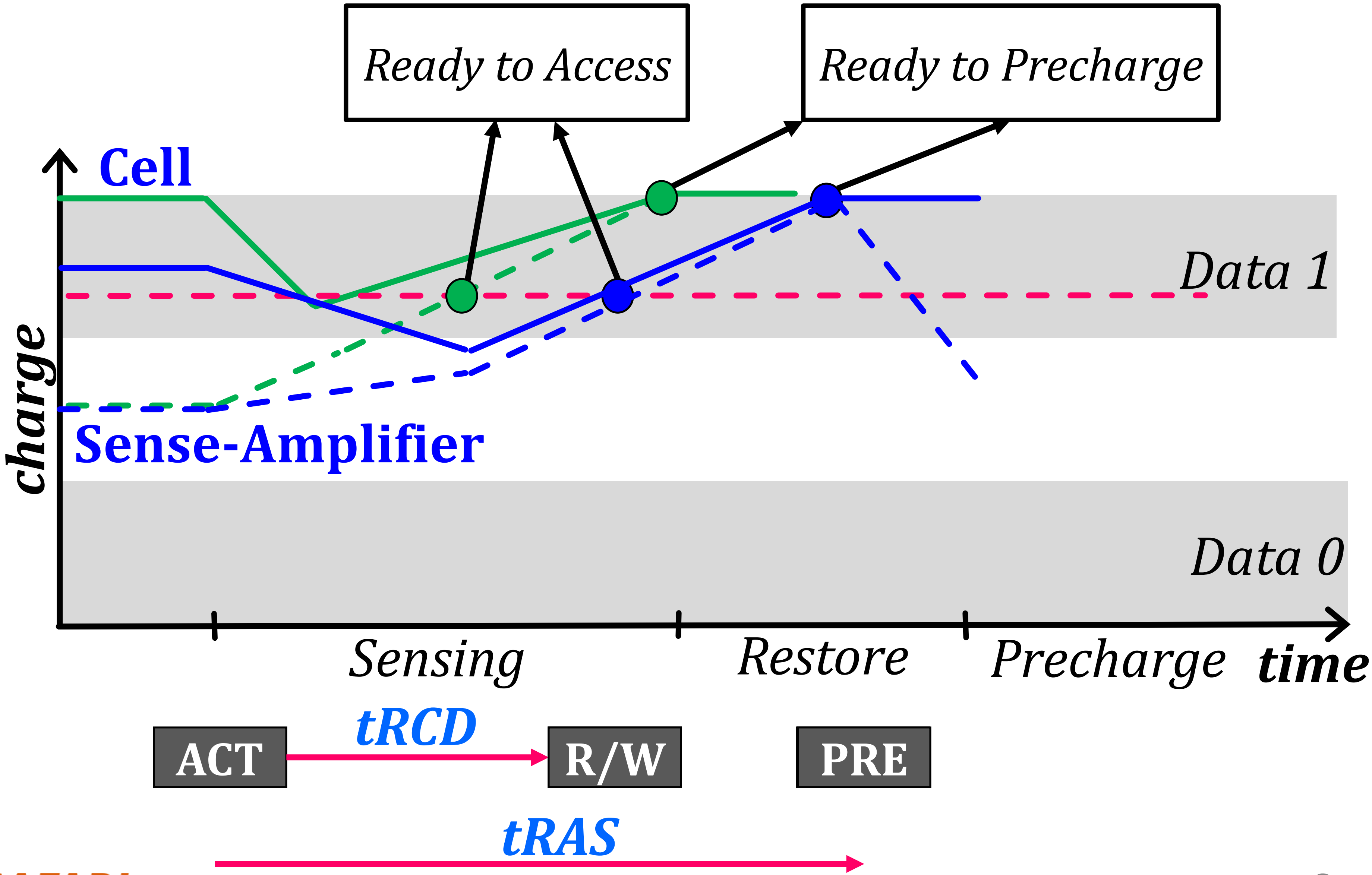
3. Row Level Temporal Locality (RLTL)

4. ChargeCache

5. Evaluation

6. Conclusion

Accessing Highly-charged Rows



Observation 1

A highly-charged DRAM row can be accessed with low latency

- tRCD: **44%**
- tRAS: **37%**

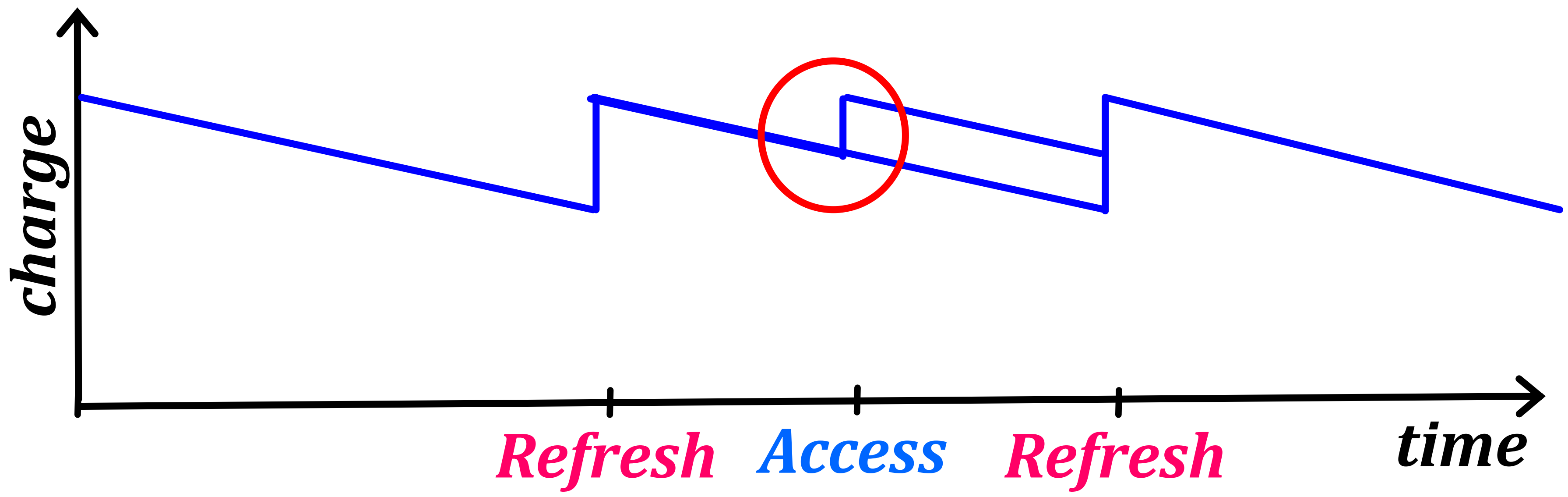
How does a row become highly-charged?

How Does a Row Become Highly-Charged?

DRAM cells **lose charge** over time

Two ways of restoring a row's charge:

- Refresh Operation
- Access



Observation 2

A row's charge is restored when the row is accessed

How likely is a recently-accessed row to be accessed again?

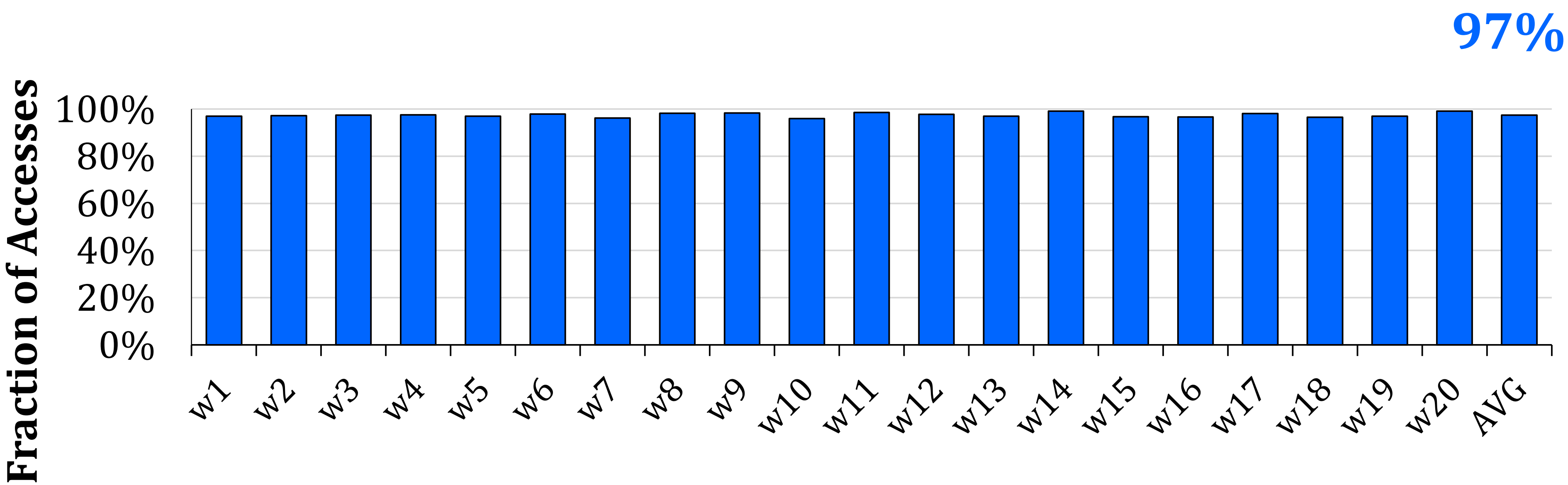
Outline

1. DRAM Operation Basics
2. Accessing Highly-charged Rows
3. **Row Level Temporal Locality (RLTL)**
4. ChargeCache
5. Evaluation
6. Conclusion

Row Level Temporal Locality (RLTL)

A **recently-accessed** DRAM row is likely to be accessed again.

- ***t***-RLTL: Fraction of rows that are accessed within time ***t*** after their previous access



8ms – RLTL for single-core workloads

8ms – RLTL for eight-core workloads

Outline

1. DRAM Operation Basics
2. Accessing Highly-charged Rows
3. Row Level Temporal Locality (RLTL)
4. **ChargeCache**
5. Evaluation
6. Conclusion

Summary of the Observations

1. A highly-charged DRAM row can be accessed with low latency

2. A row's charge is restored when the row is accessed

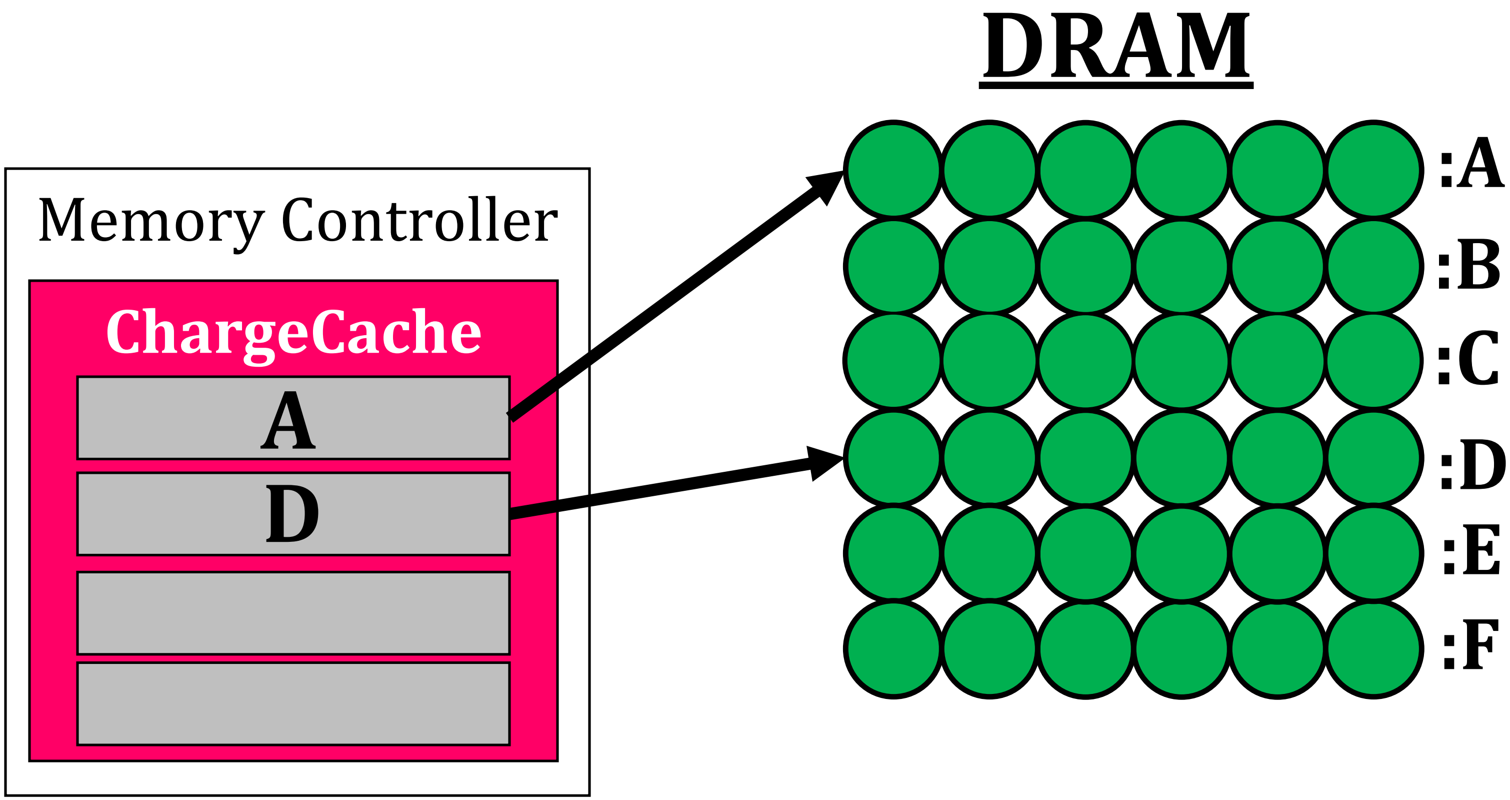
3. A **recently-accessed** DRAM row is likely to be accessed again:

Row Level Temporal Locality (RLTL)

Key Idea

Track **recently-accessed** DRAM rows and use **lower timing parameters** if such rows are accessed again

ChargeCache Overview



Requests: A D A 

ChargeCache Miss: Use Default Timings

ChargeCache Hit: Use Lower Timings

Area and Power Overhead

- **Modeled with CACTI**

- **Area**
  - ~5KB for 128-entry ChargeCache
  - 0.24% of a 4MB Last Level Cache (LLC) area

- **Power Consumption**
  - 0.15 mW on average (static + dynamic)
  - 0.23% of the 4MB LLC power consumption

Outline

1. DRAM Operation Basics
2. Accessing Highly-charged Rows
3. Row Level Temporal Locality (RLTL)
4. ChargeCache
5. **Evaluation**
6. Conclusion

Methodology

- **Simulator**
  - Ramulator *[Kim+, CAL'15]*
    https://github.com/CMU-SAFARI/ramulator
- **Workloads**
  - 22 single-core workloads
    - SPEC CPU2006, TPC, STREAM
  - 20 multi-programmed 8-core workloads
    - By randomly choosing from single-core workloads
  - Execute at least 1 billion representative instructions per core (Pinpoints)
- **System Parameters**
  - 1/8 core system with 4MB LLC
  - Default tRCD/tRAS of 11/28 cycles

Mechanisms Evaluated

Non-Uniform Access Time Memory Controller (NUAT)
[Shin+, HPCA'14]

- Key idea: Access only ***recently-refreshed*** rows with lower timing parameters
  - ➢ ***Recently-refreshed*** rows can be accessed faster
  - ➢ Only a small fraction (10-12%) of accesses go to ***recently-refreshed*** rows

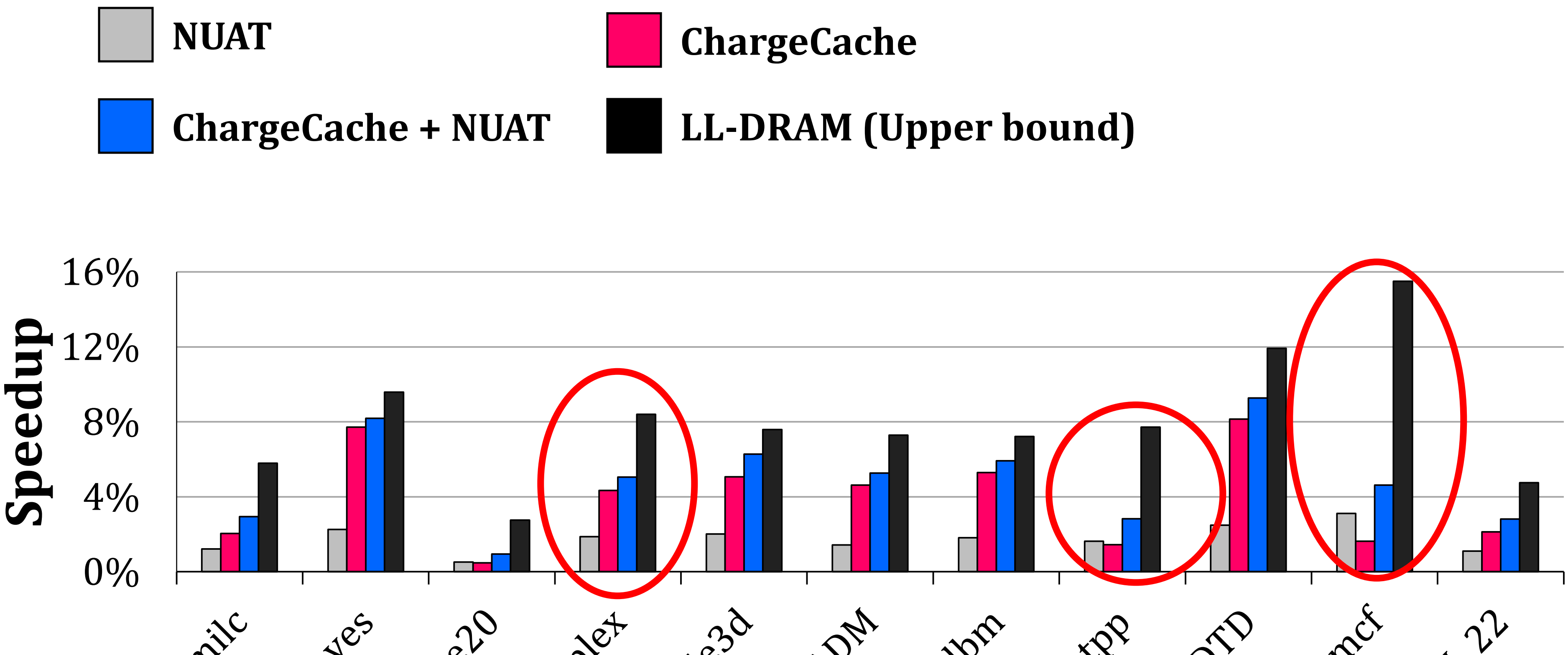
ChargeCache

- ➢ ***Recently-accessed*** *rows can be accessed faster*
- ➢ *A large fraction (86-97%) of accesses go to* ***recently-accessed*** *rows* *(**RLTL**)*
- 128 entries per core, On hit: tRCD-7, tRAS-20 cycles

Upper Bound: Low Latency DRAM

- Works as ChargeCache with 100% Hit Ratio
- On all DRAM accesses: tRCD-7, tRAS-20 cycles

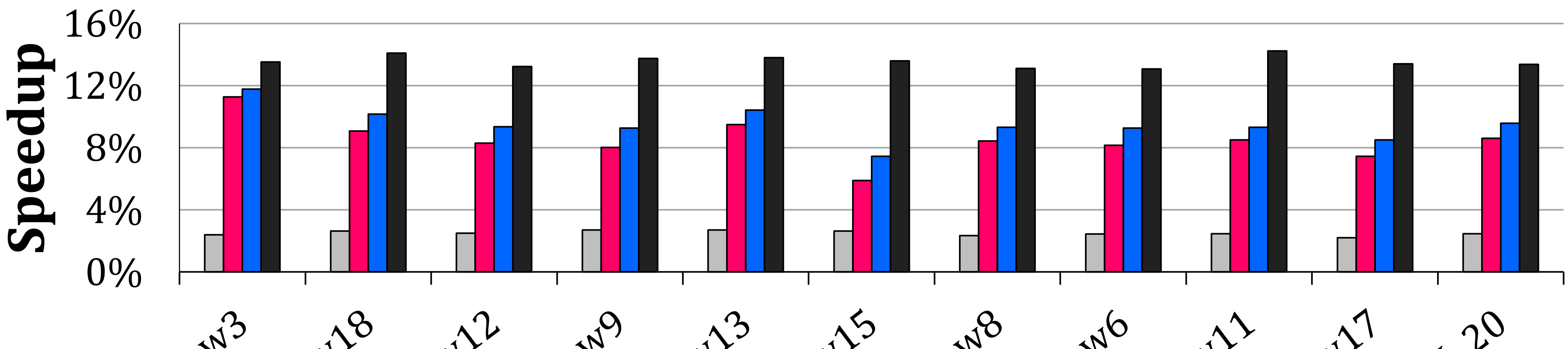
Single-core Performance



ChargeCache improves single-core performance

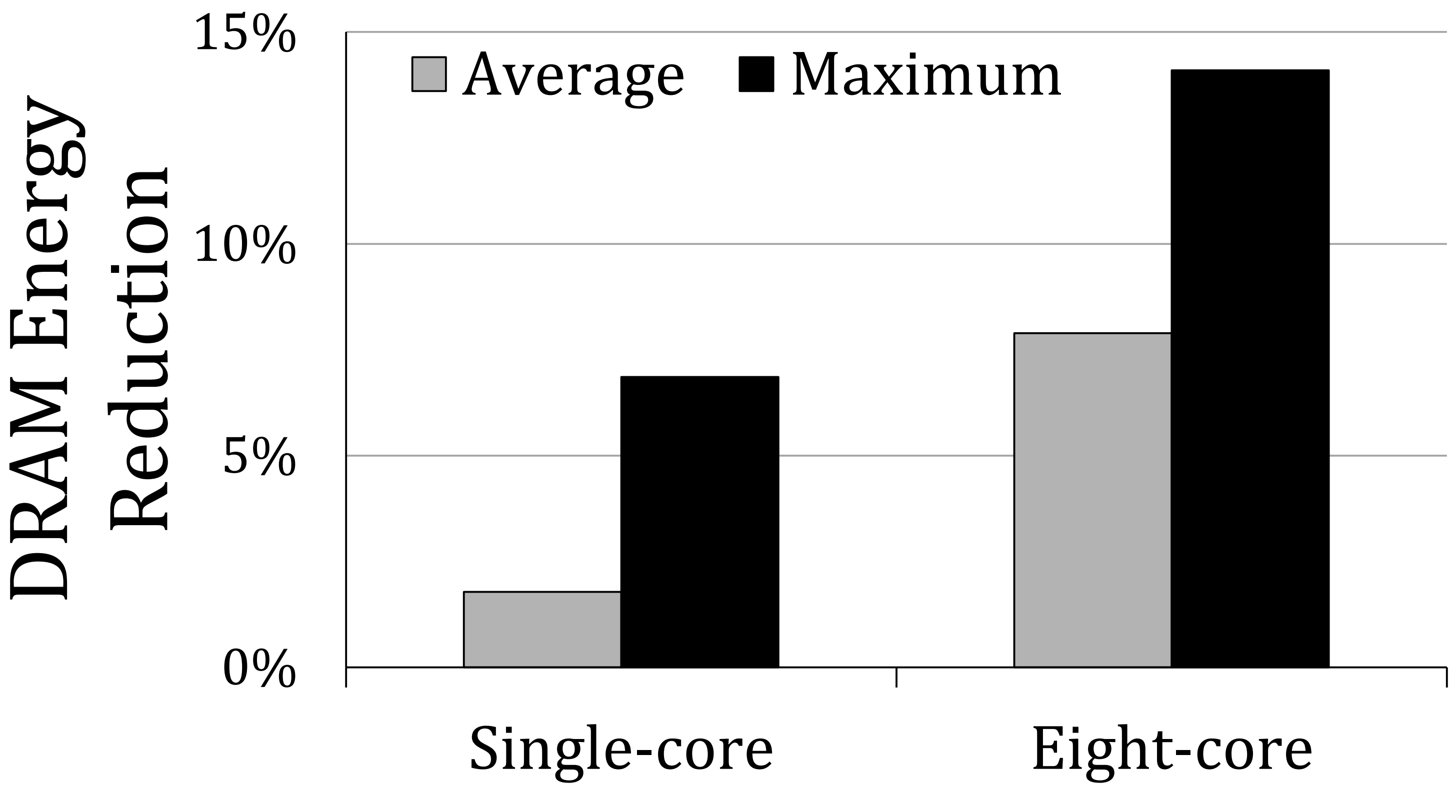
Eight-core Performance

- NUAT 2.5%
- ChargeCache 9%
- ChargeCache + NUAT
- LL-DRAM (Upperbound) 13%



ChargeCache significantly improves multi-core performance

DRAM Energy Savings



ChargeCache reduces DRAM energy

Other Results In The Paper

- Detailed analysis of the Row Level Temporal Locality phenomenon
- ChargeCache hit-rate analysis
- Sensitivity studies
  - Sensitivity to **t** in **t**-RLTL
  - ChargeCache capacity

Outline

1. DRAM Operation Basics
2. Accessing Highly-charged Rows
3. Row Level Temporal Locality (RLTL)
4. ChargeCache
5. Evaluation
6. **Conclusion**

Conclusion

- ChargeCache reduces average DRAM access latency at low cost
- **Observations**:
  1) A highly-charged DRAM row can be accessed with low latency
  2) A row's charge is restored when the row is accessed
  3) A recently-accessed row is likely to be accessed again**: Row Level Temporal Locality (RLTL)**
- **Key Idea**: Track recently-accessed DRAM rows and use lower timing parameters if such rows are accessed again
- **ChargeCache:**
  - Low cost & no modifications to the DRAM
  - Higher performance (**8.6-10.6%** on average for 8-core)
  - Lower DRAM energy (**7.9%** on average)

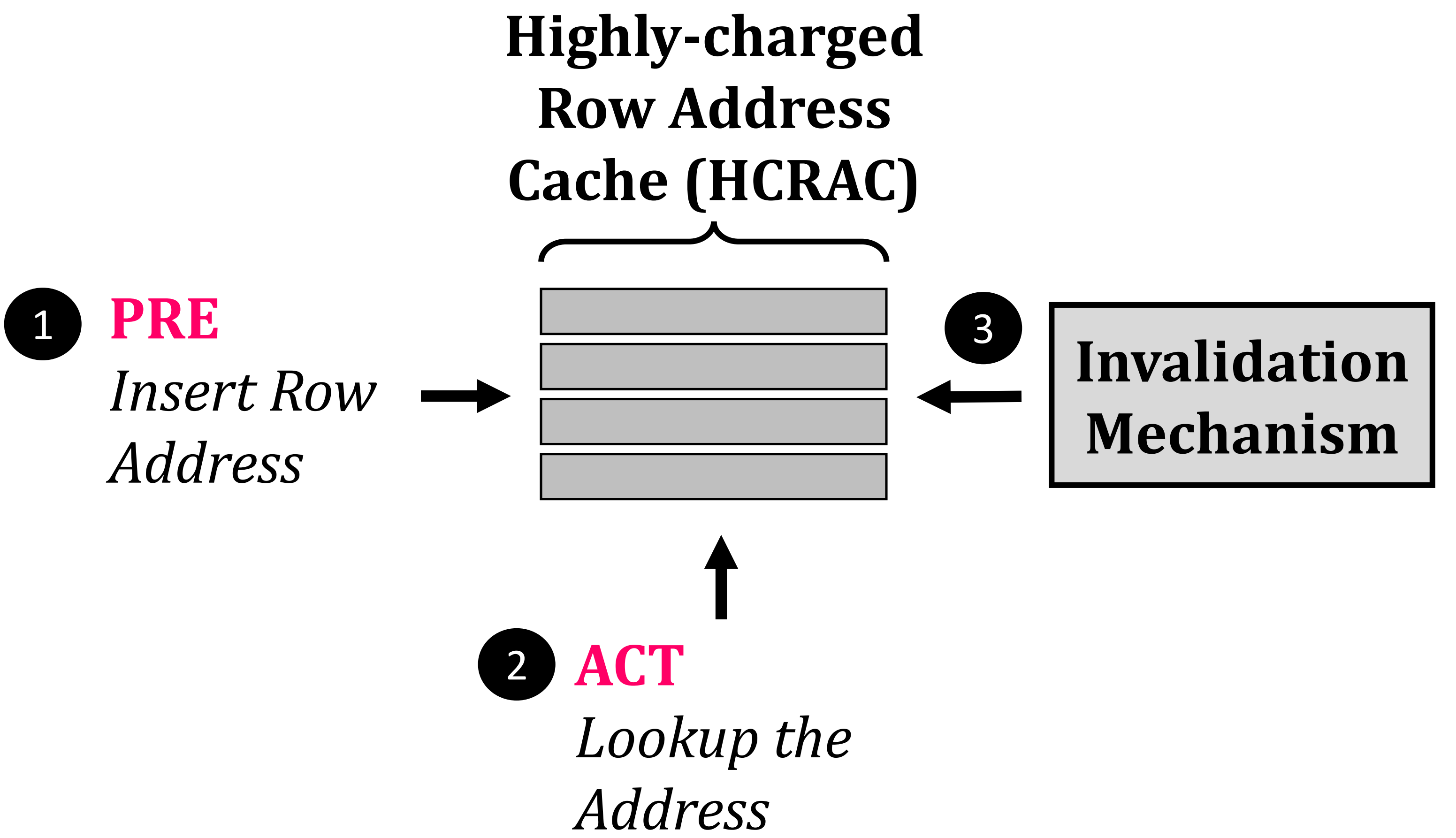
ChargeCache
Reducing DRAM Latency by Exploiting Row Access Locality

Hasan Hassan,
Gennady Pekhimenko,
Nandita Vijaykumar,
Vivek Seshadri, Donghyuk Lee,
Oguz Ergin, Onur Mutlu

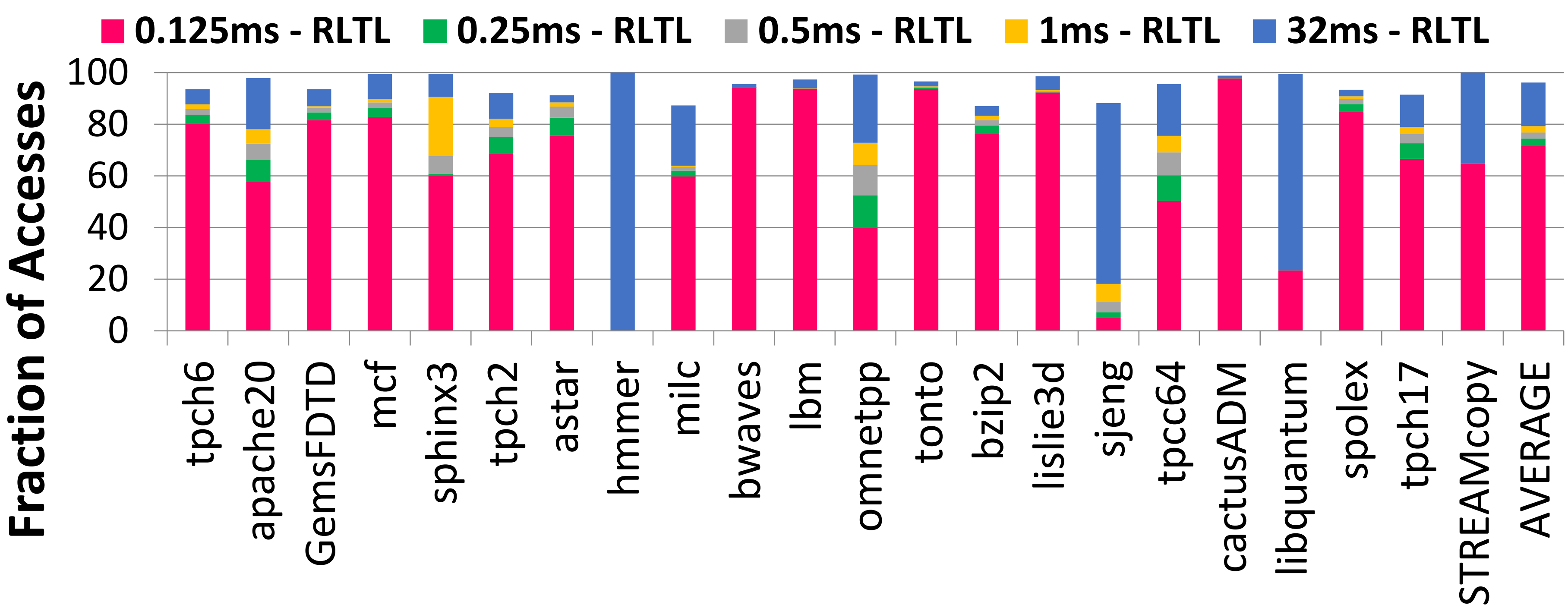
SAFARI
Carnegie Mellon
TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY
kasirga

Backup Slides

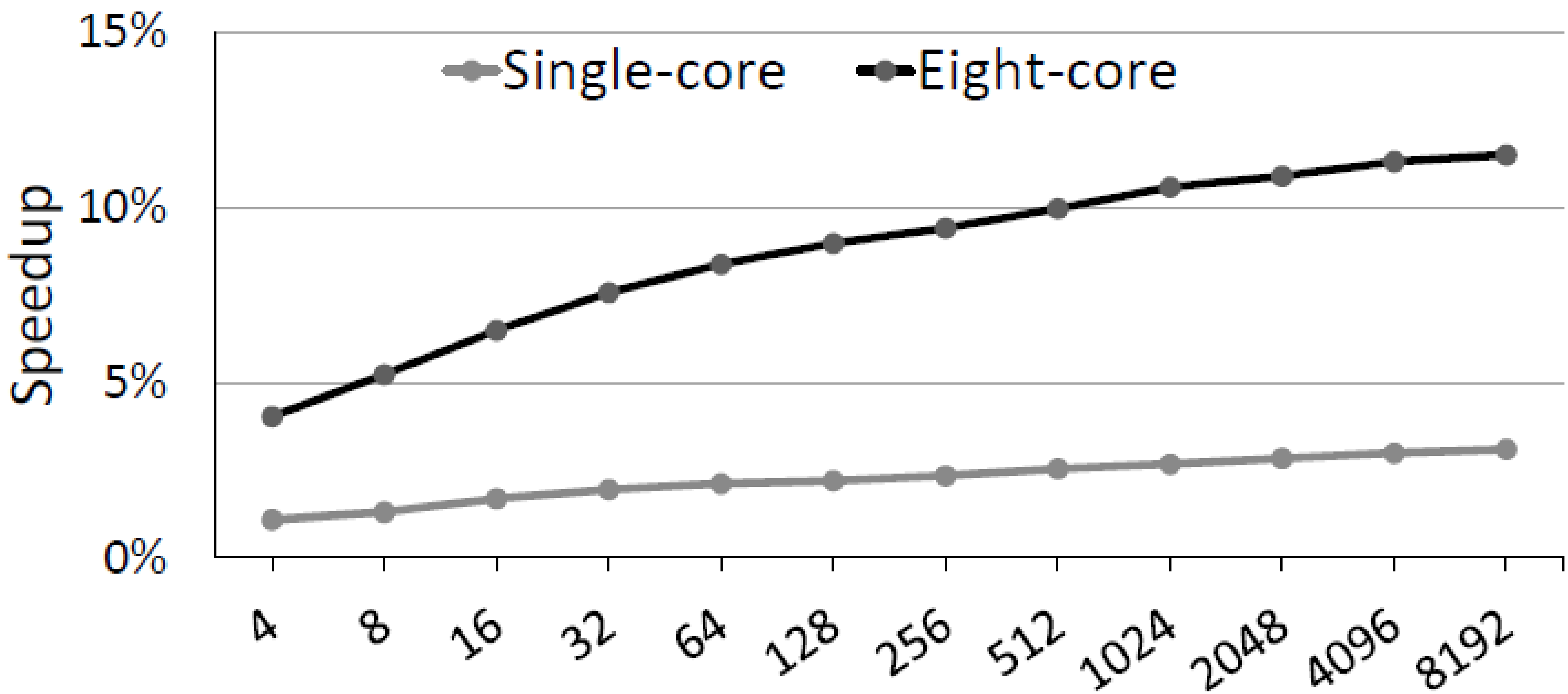
Detailed Design



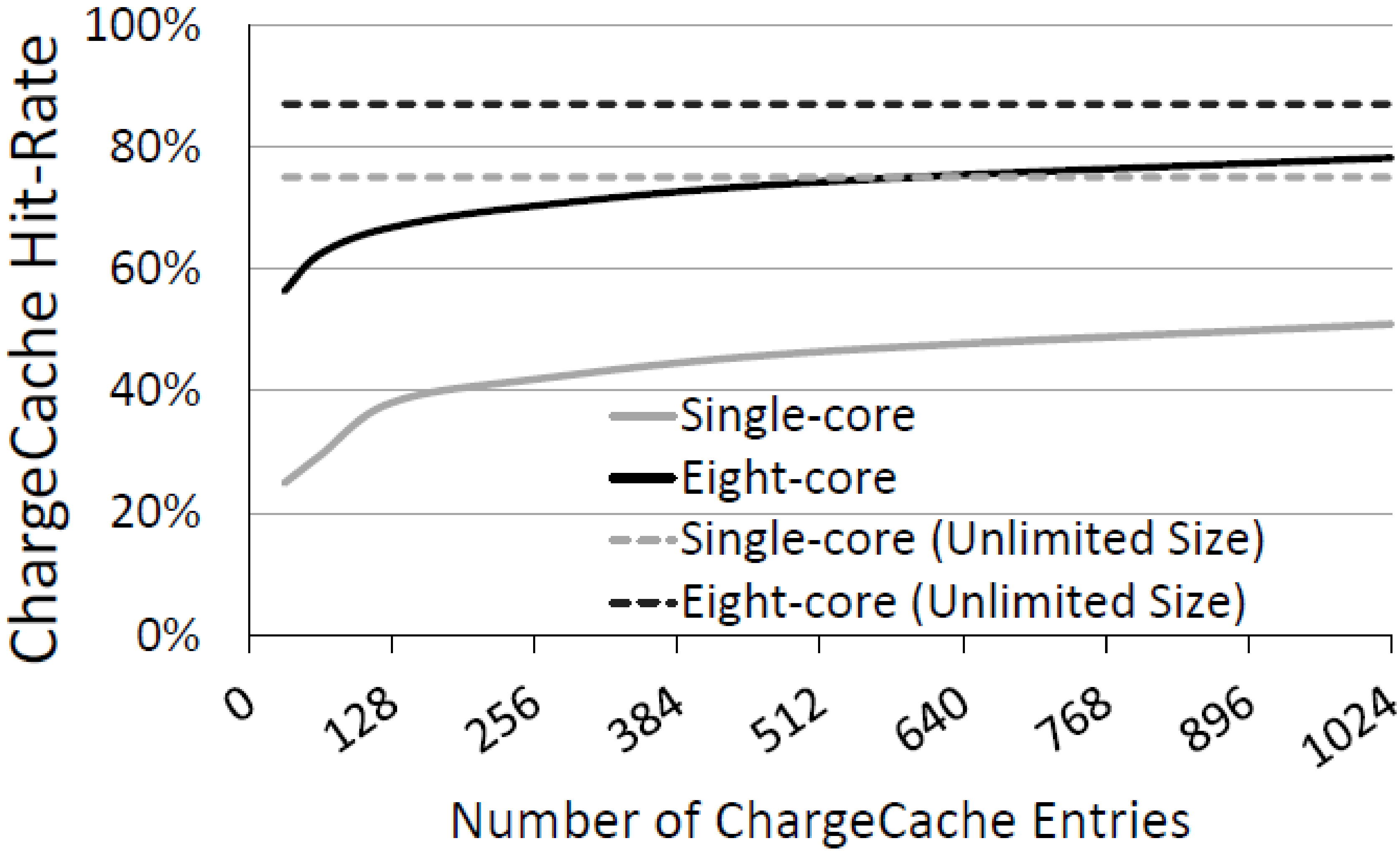
RLTL Distribution



Sensitivity on Capacity



Hit-rate Analysis



Sensitivity on t-RLTL