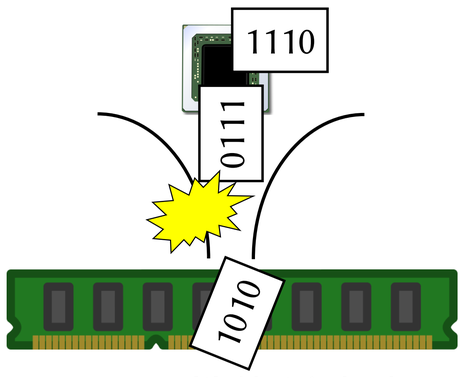
次世代メモリデバイスによるアプリケーションの自動高速化

「データを近似し出し入れを高速化」

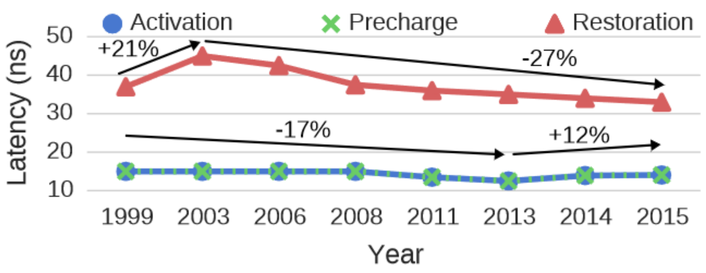
東京大学 情報理工学系研究科 創造情報学専攻
助教　穐山空道（あきやま そらみち）
akiyama@ci.i.u-tokyo.ac.jp

[解決すべき問題：メモリとCPUの性能差]

- CPUの性能は指数的増大
- メモリの性能（レイテンシ）は10年以上一定
- 具体的には
  - 足し算一回は 0.1 ns のオーダー
  - メモリアクセス一回は数十 ns のオーダー



メモリ性能が障壁
（Memory Wall）



メモリ（DRAM）の内部動作に必要な待ち時間：10年以上ほぼ一定 [1]

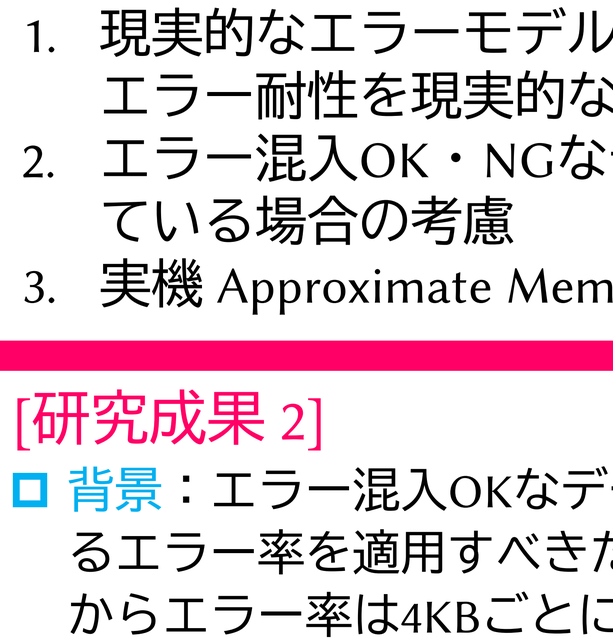
[1] Kevin K. Chang *et al.*, “Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization”, SIGMETRICS’16

- 原因
  - レイテンシ（not 帯域）は並列化で改善しない
  - キャッシュ階層を増やすとレイテンシ悪化
  - プロセスがシュリンクしても速度向上しない

メモリレイテンシを削減する抜本的な手法が必要

[未来ビジョン：Approximate Memoryの活用]

- 高速アクセスできるが、たまにビット化けが起きるメモリ
  - DRAMの内部動作の待ち時間を調整し実現*



DRAMのセルからデータを取り出す例：12.5 ns待てば確実に読み出せる電圧に達するが、7.5 nsで止めても多くのケースで正しく動作する

*他の実現方法（電圧調整等）もあるが、本方式がよく研究されている

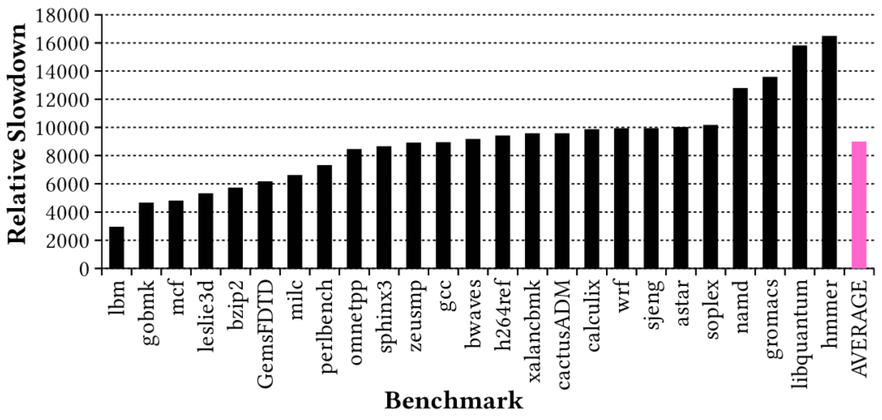
- 課題：デバイスレベルでは研究があるが、ソフトウェアからどう使えばよいか不明
  - 待ち時間をどうやって決定するのか？
  - エラー混入OK・NGな領域の区別？

[ACT-Iでの成果]

1. 現実的なエラーモデルでアプリケーションのエラー耐性を現実的な時間で評価
2. エラー混入OK・NGなデータが細粒度に混在している場合の考慮
3. 実機 Approximate Memory でのアプリ実行環境

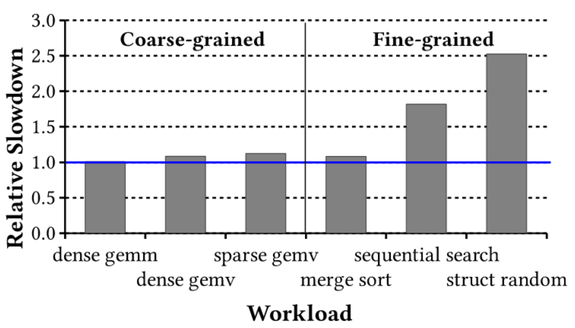
[研究成果 1]

- 背景：エラー率の適切な設定にはエラー率とアプリケーションの出力の関係が必要
  - 待ち時間とエラー率の関係は研究されている
- 課題：
  - 類似研究（信頼性等）ではエラーモデルがランダム → DRAMのエラーはランダムではない
  - ハードウェアのシミュレータは遅すぎる → AI やビッグデータ分析のアプリに適用不可能

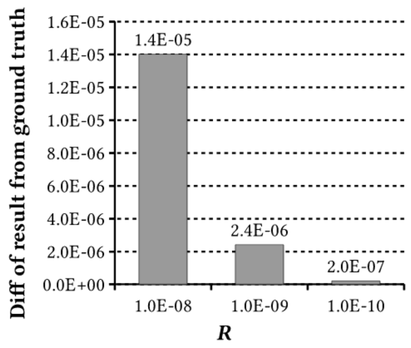


gem5 での実行時間と実機上での実行時間の比

- 提案：エラーはDRAMの内部動作の待ち時間削減で発生 → 内部動作の回数からエラー発生数を見積る
  - 内部動作の回数はパフォーマンスカウンタを用いて低オーバーヘッドで計測可能



提案システムでのマイクロベンチマークの実行時間



物理シミュレーションベンチマークの出力の誤差とエラー率の関係

Soramichi Akiyama: “A Lightweight Method to Evaluate Effect of Approximate Memory with Hardware Performance Monitors”, *IEICE Trans. on Inf. & Syst.*, Vol. E102-D, No. 12, Dec. 2019

[研究成果 2]

- 背景：エラー混入OKなデータとNGなデータに異なるエラー率を適用すべきだが、メモリの実装の仕組からエラー率は4KBごとにしか設定できない
- 課題：二種類のデータが混在していると、それらは離れた領域に配置され性能が悪化

```
struct tree_node {
   int id; // id of the node, critical
   struct tree_node *r; // pointer to the right child, critical
   struct tree_node *l; // pointer to the left child, critical
   double score; // score of this node, approximate
};

int size = 1000 * sizeof(struct tree_node);
struct tree_node *nodes = malloc(size);
```

r, l, score がアクセス局所性を持つとき、これらを 4KB 離れた場所に配置すると性能劣化

- 貢献：
  - 現実的なベンチマークを分析し、2種のデータが混在することが実際にあると確認（課題の発見）
  - 分離したものを同時フェッチする手法の予備実験

Name	C1	C2	C3
milc	Y	N	Y
sjeng	Y	N	N
libquantum	Y	N	Y
lbm	N	-	-
omnetpp	Y	N	N
soplex	Y	N	N
gobmk	Y	Y	N
gcc	Y	N	N
mcf	Y	Y	N
dealII	N	-	-
namd	Y	N	Y
Graph 500	N	-	-
GraphMat	N	-	-

SPEC CPU 2006 他の分析結果

C1: 最もキャッシュミスが多いデータは構造体またはクラスか？

C2: その構造体にポインタとそれ以外が混在しているか？

C3: その構造体に浮動小数点数が含まれるか？

→ 多くのベンチマークでエラー混入OKなデータとNGなデータが混在する可能性がある

Soramichi Akiyama: “Assessing Impact of Data Partitioning for Approximate Memory in C/C++ Code”, *The 10th Workshop on Systems for Post-Moore Architectures (SPMA)*, Apr. 2020
穐山空道, 塩谷亮太: “Approximate Memory のデータ分離に起因する性能低下を抑制するプリフェッチ手法”, *ETNET 2019*（IEICE CPSY 研究会優秀若手発表賞）

[研究成果 3] 実機Approximate Memory上でのアプリ実行を可能に（ETH Zurichと協業、論文出版後公表）