The Hardness of Data Packing

Rahman Lavaee

University of Rochester

POPL 2016, St. Petersburg, FL, USA

January 20, 2016
Figure 1: Program access pattern example (adopted from [Kleen et al., 2007])
Poor Spatial Locality

Figure 2: Program access pattern example (adopted from [Kleen et al., 2007])
Question 1
Are we getting enough of spatial locality?
Spatial Locality

**Question 1**
Are we getting enough of spatial locality?

**Question 2**
Given a fixed-size cache, how does the performance change as we increase the cache block size?
Spatial Locality

Question 1
Are we getting enough of spatial locality?

Question 2
Given a fixed-size cache, how does the performance change as we increase the cache block size?

![Graph showing the relationship between cache block size and miss rate. The graph indicates a decreasing trend in miss rate as cache block size increases.]
Spatial Locality

**Question 1**
Are we getting enough of spatial locality?

**Question 2**
Given a fixed-size cache, how does the performance change as we increase the cache block size?

“When increasing the cacheline size from 64 bytes to 128 bytes, the number of misses will be reduced by 10%-30%, depending on the workload. However, because of the low cacheline utilization, this more than doubles the amount of unused traffic to the L1 caches. [Laer et al., 2015]."
Spatial Locality

Question 1
Are we getting enough of spatial locality?

Question 2
Given a fixed-size cache, how does the performance change as we increase the cache block size?

When increasing the cacheline size from 64 bytes to 128 bytes, the number of misses will be reduced by 10%-30%, depending on the workload. However, because of the low cacheline utilization, this more than doubles the amount of unused traffic to the L1 caches. [Laer et al., 2015].
Question 1
Are we getting enough of spatial locality? Maybe not

Question 2
Given a fixed-size cache, how does the performance change as we increase the cache block size?

When increasing the cacheline size from 64 bytes to 128 bytes, the number of misses will be reduced by 10%-30%, depending on the workload. However, because of the low cacheline utilization, this more than doubles the amount of unused traffic to the L1 caches. [Laer et al., 2015].
Definition (Cache-Conscious Data Placement)

Given a sequence of $N$ references to fixed-size objects, and all the details of the cache, find a placement that minimizes the cache misses.
Definition (Cache-Conscious Data Placement)

Given a sequence of $N$ references to fixed-size objects, and all the details of the cache, find a placement that minimizes the cache misses.

Prior work by Petrank and Rawtiz [Petrank and Rawitz, 2002] shows that this problem is NP-hard to approximate even within a ratio of $O(N^{1-\epsilon})$, for every $\epsilon > 0$. 
Petrank’s hardness result is based on \textit{conflict misses}, implying that it is very hard to minimize those misses.
Petrank’s hardness result is based on conflict misses, implying that it is very hard to minimize those misses.

Spatial locality does not have much to do with conflict misses.
Petrank’s hardness result is based on **conflict misses**, implying that it is very hard to minimize those misses.

Spatial locality does not have much to do with conflict misses.

Let us assume a fully-associative conflict-free cache.
Conflict vs. Capacity

- Are fully-associative caches used in practice?
Are fully-associative caches used in practice?

- virtual memory systems, Memcached [Fitzpatrick, 2004], caching for embedded devices [Huneycutt et al., 2002].
- software-managed hardware caches
Conflict vs. Capacity

The performance of today's caches on common workloads is mostly determined by their capacity.
The performance of today's caches on common workloads is mostly determined by their capacity.

Example

Cantin and Hill [Cantin and Hill, 2003] report average miss rates for three 256KB LRU caches with different associativities, on SPEC2000.

<table>
<thead>
<tr>
<th>Associativity</th>
<th>Miss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully associative</td>
<td>0.0163969</td>
</tr>
<tr>
<td>8-way set associative</td>
<td>0.0167211</td>
</tr>
<tr>
<td>4-way set associative</td>
<td>0.0167639</td>
</tr>
</tbody>
</table>

Capacity misses constitute 98% of misses for both the 8-way and 4-way set associative caches.
For a fully-associative cache, data layout is irrelevant if data object size is \textbf{the same} as the block size, unless objects are free to be non-aligned.
Object Size vs. Cache Block Size

- For a fully-associative cache, data layout is irrelevant if data object size is **the same** as the block size, unless objects are free to be non-aligned.

- But data layout is important when data object size is either **smaller** than, or **not a multiple** of the block size.
Data Packing

- Data placement for a fully associative cache:

**Definition (Data Packing)**

Given a sequence of $N$ references to objects of size $B/p$, where $B$ is the cache block size and $p$ (packing factor) is an integer no smaller than 2, find a placement (equivalently, a packing) of objects in memory that minimizes the cache misses.
We prove that for every packing factor starting at $2$,

- for every number of blocks in cache starting at $5$, this problem is NP-hard to approximate within a factor $O(N^{1-\epsilon})$, for every $\epsilon > 0$.
We prove that for every packing factor starting at 2,

- for every number of blocks in cache starting at 5, this problem is
  NP-hard to approximate within a factor $O(N^{1-\epsilon})$, for every $\epsilon > 0$.
- Our result holds for all sensible replacement policies.
We prove that for every packing factor starting at 2,
for every number of blocks in cache starting at 5, this problem is NP-hard to approximate within a factor $O(N^{1-\varepsilon})$, for every $\varepsilon > 0$. Our result holds for all sensible replacement policies.

Definition ($q$-sensible replacement policy, adapted from [Petrank and Rawitz, 2002])

\[
\sigma^\ell \text{ incurs } \begin{cases} 
\leq m + q \text{ misses} & \text{if objects in } \sigma \text{ are packed in a total of at most } m \text{ blocks} \\
\geq \ell \text{ misses} & \text{otherwise}
\end{cases}
\]

($m$: number of blocks in cache)
We prove that for every packing factor starting at 2,
- for every number of blocks in cache starting at 5, this problem is NP-hard to approximate within a factor $O(N^{1-\epsilon})$, for every $\epsilon > 0$.
- Our result holds for all sensible replacement policies.

**Definition (q-sensible replacement policy, adapted from [Petrank and Rawitz, 2002])**

\[ \sigma^\ell \text{ incurs} \begin{cases} 
  \leq m + q \text{ misses} & \text{if objects in } \sigma \text{ are packed in a total of at most } m \text{ blocks} \\
  \geq \ell \text{ misses} & \text{otherwise} 
\end{cases} \]

(m: number of blocks in cache)

- All well-known replacement policies are 0-sensible: FIFO, LRU, MRU, etc.
Proof Strategy

- Reduction from 3SAT.
Proof Strategy

- Reduction from **3SAT**.

- Given a **3SAT** instance on a set \( \mathcal{V} \) of variables and a collection \( \mathcal{C} \) of clauses, our reduction ensures that

\[
\text{Miss}_{\text{opt}} \leq \text{Miss}_{\text{low}} = (m + q)(6|\mathcal{C}| + |\mathcal{V}| + 4) = o(N^{\epsilon/2})
\]

if

\[
\text{Miss}_{\text{high}} \geq |\mathcal{C}| + |\mathcal{V}|)^{4^{-\epsilon}} - 1 = \omega(N^{1-\epsilon/2})
\]

mapping from a positive **3SAT** instance.

Any approximation algorithm with a factor at least as good as \( O(N^{1-\epsilon/2}) \) would place **3SAT** in \( P \).
Proof Strategy

- Reduction from 3SAT.
- Given a 3SAT instance on a set $\mathcal{V}$ of variables and a collection $\mathcal{C}$ of clauses, our reduction ensures that

$$Miss_{opt} \leq Miss_{low} = (m + q)(6|\mathcal{C}| + |\mathcal{V}| + 4) = o(N^{\epsilon/2})$$

if mapping from a positive 3SAT instance.

$$Miss_{high} = (|\mathcal{C}| + |\mathcal{V}|)^{4\epsilon - 1} = \omega(N^{1 - \epsilon/2})$$

- Any approximation algorithm with a factor at least as good as $O(N^{1-\epsilon})$ would place 3SAT in P.
Given a 3SAT instance on a set $\mathcal{V}$ of variables and a collection $\mathcal{C}$ of clauses,
Given a 3SAT instance on a set $V$ of variables and a collection $C$ of clauses,

for every variable $u \in V$, we introduce $3p$ data objects:

$c_1^u$  \hspace{1cm}  $c_2^u$  \hspace{1cm}  $\cdots$  \hspace{1cm}  $c_p^u$

$b_1^u$  \hspace{1cm}  $b_2^u$  \hspace{1cm}  $\cdots$  \hspace{1cm}  $b_p^u$

$a_1^u$  \hspace{1cm}  $a_2^u$  \hspace{1cm}  $\cdots$  \hspace{1cm}  $a_p^u$
Given a 3SAT instance on a set $V$ of variables and a collection $C$ of clauses,

for every variable $u \in V$, we introduce $3p$ data objects:

We ensure that if mapping from a positive 3SAT instance, then the optimal data packing solution packs these objects in one of the two ways:

\begin{align*}
    a_1^u & \quad a_2^u & \quad \cdots & \quad a_p^u \\
    b_1^u & \quad b_2^u & \quad \cdots & \quad b_p^u \\
    c_1^u & \quad c_2^u & \quad \cdots & \quad c_p^u
\end{align*}

associated with $u = \text{True}$

\begin{align*}
    a_1^u & \quad a_2^u & \quad \cdots & \quad a_p^u \\
    b_1^u & \quad b_2^u & \quad \cdots & \quad b_p^u \\
    c_1^u & \quad c_2^u & \quad \cdots & \quad c_p^u
\end{align*}

associated with $u = \text{False}$
Construction Details: 3SAT Clauses

- For every typical clause $c = x \lor \neg y \lor \neg z$, we add the sequence

$$\left( a^x_1 \circ c^y_1 \circ c^z_1 \circ a^x_2 \circ \ldots \circ a^x_p \circ a^y_2 \circ \ldots \circ a^y_p \circ a^z_2 \circ \ldots \circ a^z_p \circ \delta \right)^{Miss_{high}},$$

- where $\delta$ is a sequence over $p(m - 5)$ objects that we ensure are packed together if mapping from a positive 3SAT instance.
How does approximability change if we use hit maximization as the objective value.
Cache Hit Maximization

- How does approximability change if we use hit maximization as the objective value.
- For a one-block cache, we can formulate the problem as a **GRAPH PACKING** problem:
  - partitioning the **proximity graph** into sets of bounded size, to maximize the weight of the edges inside partitions
* For general number of blocks in cache \((m)\), we formulate the problem as a **Hypergraph Packing** problem.
For general number of blocks in cache \((m)\), we formulate the problem as a **Hypergraph Packing** problem.

**Definition (Maximum Weighted \(p\)-Set Packing [Arkin and Hassin, 1997] (Hypergraph Packing))**

Given a collection \(\mathcal{F}\) of weighted sets, each containing at most \(p\) elements, find a collection of disjoint sets in \(\mathcal{F}\) which has the maximum total weight.
For general number of blocks in cache \((m)\), we formulate the problem as a **Hypergraph Packing** problem.

**Definition** (**Maximum Weighted \(p\)-Set Packing** [Arkin and Hassin, 1997] (Hypergraph Packing))

Given a collection \(\mathcal{F}\) of weighted sets, each containing at most \(p\) elements, find a collection of disjoint sets in \(\mathcal{F}\) which has the maximum total weight.

A simple greedy approach gives a \(p\)-factor algorithm for this problem.
Assigning the Hyperedge Weights

Example

Consider the following reference sequence, for a 2-block cache and a packing factor of 3.

\[ a \ u \ v \ b \ c \ v \ b \]

How to set the weight of \( \{a, b, c\} \)?
Assigning the Hyperedge Weights

Example

Consider the following reference sequence, for a 2-block cache and a packing factor of 3.

\[
\{a, b, c\}
\]

How to set the weight of \(\{a, b, c\}\)?

- Packing \(a, b,\) and \(c\) in the same block will potentially give three block reuses.
- However, the first block reuse may only hit in cache, if \(u\) is placed in the same block as \(v\).
Assigning the Hyperedge Weights

Example
Consider the following reference sequence, for a 2-block cache and a packing factor of 3.

\[
\{a, u, v, b, c, v, b\}
\]

How to set the weight of \(\{a, b, c\}\)?

Idea
We consider only the block reuses that the packing guarantees, independently from other packings. \(\rightarrow\) proximity hypergraph
Assigning the Hyperedge Weights

Example

Consider the following reference sequence, for a 2-block cache and a packing factor of 3.

How to set the weight of \{a, b, c\}?

Idea

We consider only the block reuses that the packing guarantees, independently from other packings. → proximity hypergraph

- The optimal solution for the original problem is within a factor \(m\) of the optimal packing for the proximity hypergraph.
Assigning the Hyperedge Weights

Example
Consider the following reference sequence, for a 2-block cache and a packing factor of 3.

How to set the weight of \(\{a, b, c\}\)?

Idea
We consider only the block reuses that the packing guarantees, independently from other packings. → proximity hypergraph

- The optimal solution for the original problem is within a factor \(m\) of the optimal packing for the proximity hypergraph.
- An approximation factor of \(mp\) for cache hit maximization
Future Work

- Restrict the phase behavior
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\epsilon)$ phases.
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\epsilon)$ phases.
  - Thus hardness holds even when restricted to those instances.

How about when the number of phases follows a smaller function: constant, logarithmic, poly-logarithmic. Can we improve the running time of the cache hit maximization algorithm? How does it perform in practice?

Data packing for partitioned cache: Need to consider coherence misses.
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\epsilon)$ phases.
  - Thus hardness holds even when restricted to those instances.
  - How about when number of phases follows a smaller function: constant, logarithmic, poly-logarithmic.
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\epsilon)$ phases.
  - Thus hardness holds even when restricted to those instances.
  - How about when number of phases follows a smaller function: constant, logarithmic, poly-logarithmic.
- Restrict to instances for which $Miss_{opt} \geq t(N)$ for some function $t$. 
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\varepsilon)$ phases.
  - Thus hardness holds even when restricted to those instances.
  - How about when number of phases follows a smaller function: constant, logarithmic, poly-logarithmic.
- Restrict to instances for which $Miss_{opt} \geq t(N)$ for some function $t$.
- Can we improve the running time of the cache hit maximization algorithm? How does it perform in practice?
Future Work

- Restrict the phase behavior
- Assume a very simple definition of phase: a loop over distinct objects.
  - Our reduction maps to reference sequences with $\Theta(N^\epsilon)$ phases.
  - Thus hardness holds even when restricted to those instances.
  - How about when number of phases follows a smaller function: constant, logarithmic, poly-logarithmic.
- Restrict to instances for which $Miss_{opt} \geq t(N)$ for some function $t$.
- Can we improve the running time of the cache hit maximization algorithm? How does it perform in practice?
- Data packing for partitioned cache: Need to consider coherence misses.
Data packing is a challenging and important problem for today's block-based, highly associative caches.
Data packing is a challenging and important problem for today's block-based, highly associative caches.

We formally defined this problem for an arbitrary number of cache blocks and packing factor.
Data packing is a challenging and important problem for today's block-based, highly associative caches.

We formally defined this problem for an arbitrary number of cache blocks and packing factor.

We showed that even for a 5-block cache, an approximate data packing solution cannot be efficiently computed, unless $P = NP$.

We showed that when aiming at hit maximization, the data packing problem admits a constant factor approximation algorithm.


Thank you