



## Improving Memory Hierarchy Performance for Irregular Applications Using Data and Computation Reorderings

By: John Mellor-Crummey, David Whalley, Ken Kennedy

Presented By: Saeed Alaei



## Outline

- Memory hierarchies and performance.
- Data and computation reordering in general.
- Regular vs. Irregular applications.
- Space filling curves:
  - Hilbert curve
  - Morton curve
- Data and computation reordering approaches.
- Experimental results

## Memory performance

- Gap between CPU speed and memory speed has increased rapidly.
- Multi-level memory hierarchies are used to address this memory access bottleneck.
- Irregular applications underutilize the memory hierarchies.
- Data and computation reordering can greatly help.

## Data reordering

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    for(k=0;k<n;k++)
      C[i][j]+=A[i][k]*B[k][j];
```

---

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    for(k=0;k<n;k++)
      C[i][j]+=A[i][k]*BT[j][k];
```

## Computation reordering

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    for(k=0;k<n;k++)
      C[i][j]+=A[i][k]*B[k][j];
```

---

```
for(i=0;i<n;i++)
  for(k=0;k<n;k++)
    for(j=0;j<n;j++)
      C[i][j]+=A[i][k]*B[k][j];
```

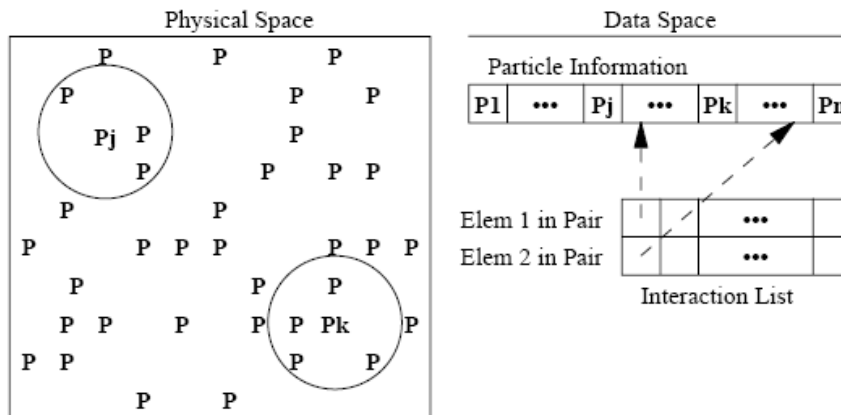
## Regular vs. Irregular Applications

- Regular applications:
  - Static analysis and fixed access pattern.
  - Techniques:
    - Loop blocking.
    - Data prefetching.
  - Instances:
    - Matrix multiplication/inversion.
- Irregular applications:
  - Dynamic analysis, access pattern unknown until runtime.
  - Proposed techniques:
    - Dynamic data reordering before major computation phase.
    - Computations reordering by blocking.

# Irregular applications

- Poor spatial and temporal locality.
- Bandwidth and latency problems:
  - Cache miss.
  - TLB miss.
- Instances:
  - N-Body simulation.

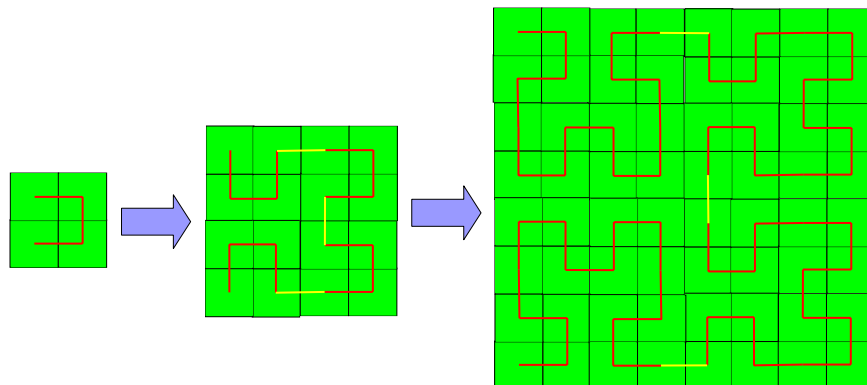
# N-Body



## Space Filling Curve

- A space-filling curve for some finite space of  $d$  dimensions ( $d \geq 2$ ) is a continuous, non-smooth curve that passes arbitrarily close to every point.
- Each point in a  $d$ -dimensional space can be mapped to the nearest position along a 1-dimensional spacefilling curve by applying a sequence of bit-level logical operations to its  $d$ -dimensional coordinates

## Hilbert Curves

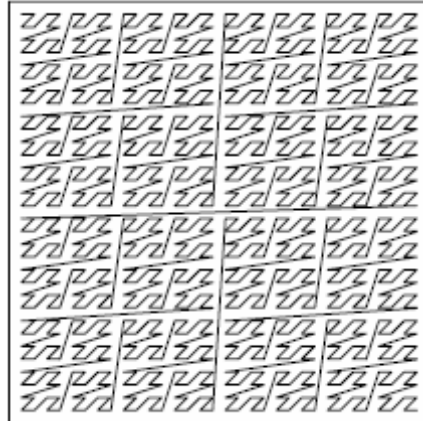


## Morton Curves

X=1010101

Y=1010010

M=11001100011001



## Reordering

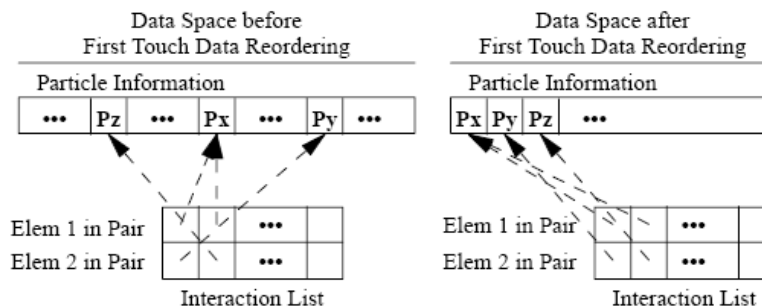
- Data Reordering
- Computation Reordering

## Data Reordering Approaches

- Changes the order of the elements of data.
- Doesn't change the order in which the elements are referenced
- Approaches:
  - First touch reordering.
  - Space filling curve reordering.

## First Touch Data Reordering

- It is a greedy approach to improve spatial localities in irregular applications.
- Changes the order of particle information
- Improves the locality in time by reordering the particle list.



## First Touch Reordering (cont'd)

- Advantages:

- Simple to implement.
- Requires linear time.

- Disadvantages:

- Interaction list must be known in advance.

## Space Filling Curve Data Reordering

- Steps:

- Particle coordinates must be mapped to positions in space filling curve.
- Particles are sorted in ascending order by their position on the curve.

- Advantages:

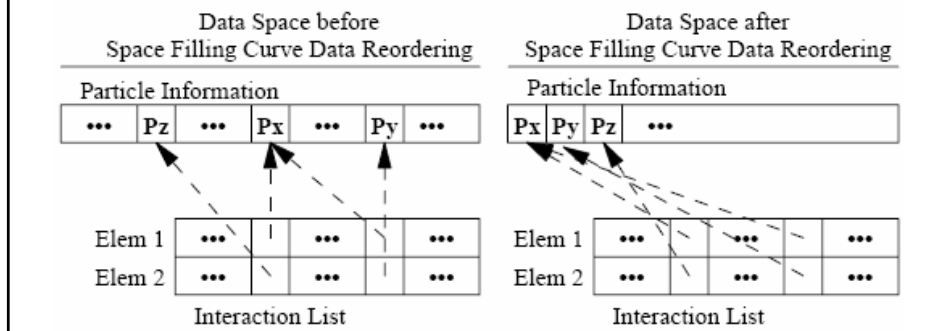
- Can be done prior to knowing the order of computation.
- Allows some computation reordering (if computed before the interaction list).

- Disadvantages:

- May require more overhead due to sorting.

## Space Filling Curve Data Reordering (cont'd)

- Increases the spatial locality.

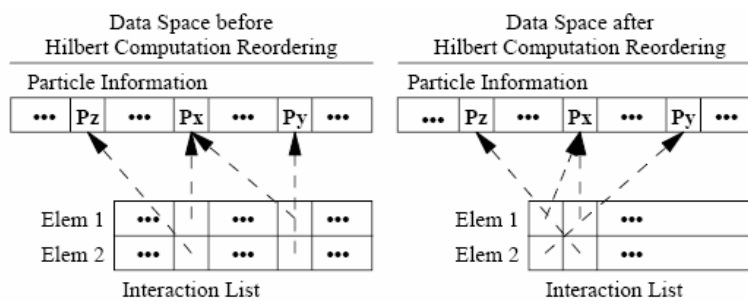


## Computation Reordering

- Changes the order in which the elements are referenced.
- Doesn't Change the order of the elements of data.
- Can improve both temporal and spatial locality of access.
- Approaches:
  - Space filling curve reordering.
  - Reordering by blocking.

## Space Filling Curve Computation Reordering

- Changes the order of elements in the interaction list.
- The order of particle information remains unchanged.
- Improves temporal locality.



## Computation Reordering by Blocking

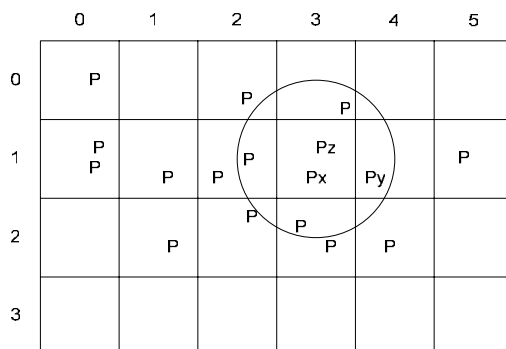
```
FOR i = 1 to number of particles DO
  FOR j in the set particles_that_interact_with[i] DO
    process interaction between particles i and j
```



```
FOR i = 1 to number of blocks of particles DO
  FOR j = i to number of blocks of particles DO
    process interactions between all interacting
    particle pairs with the first particle in block i
    and the second in block j
```

## Computation Reordering by Blocking (cont'd)

- Blocking can also be done based on different approaches (e.g. by taking the address).

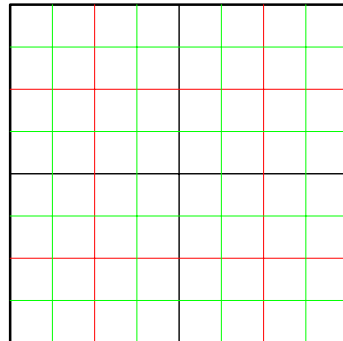


## Computation Reordering by Blocking (cont'd)

- Blocking can be extended to multiple levels.
- A blocking factor depends on architectural characteristics:
  - Size of cache lines or page size for TLB.
  - The number of sets in the cache.
  - Associativity.
  - Replacement policy.
  - ...

## Computation Reordering by Blocking (cont'd)

- The recursive divide and conquer approach can be used to achieve a machine independent blocking.



## Computation Reordering by Blocking (cont'd)

- For each pair  $[P_i, P_j]$  in the interaction list, we compute the interleaving of  $[\text{block\_of}(P_i), \text{block\_of}(P_j)]$ .
- This is equivalent to using a Morton curve.
- A Hilbert curve can also be used.
- Both Hilbert and Morton curve give a recursive blocking.

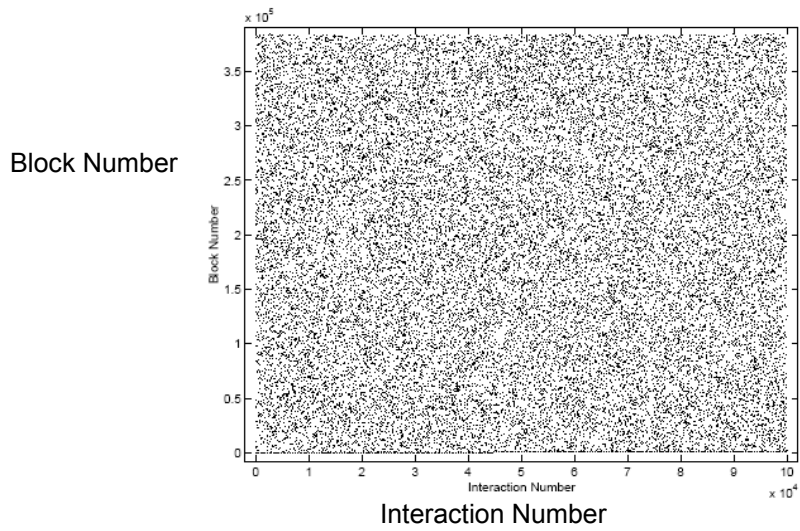
## Applying The Techniques

- Particle codes:
  - Moldyn (a synthetic benchmark).
  - Magi
  - CHAD

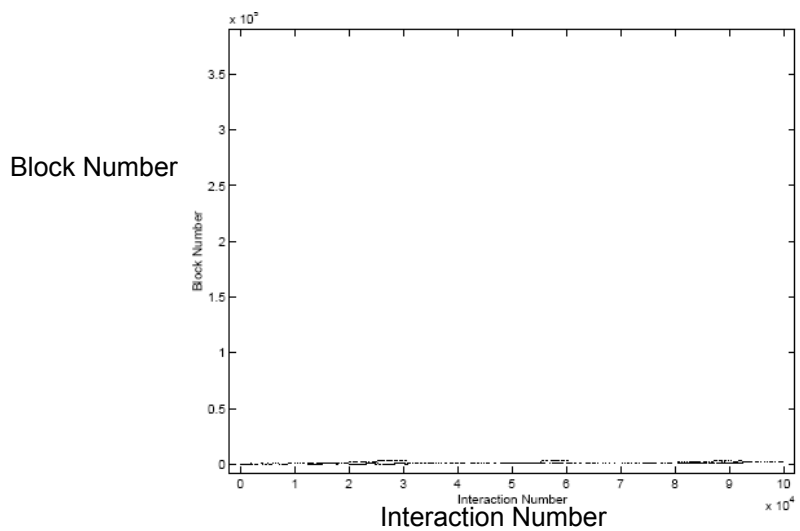
## Moldyn

- A synthetic benchmark for molecular dynamics simulation.
- Closely resembles the N-Body example.
- The time consuming part is the computeForces function called on list of interactions.
- The experiment:
  - Performed SGI O2 based on R10000 MIPS.
  - 256,000 particles, over 27 million interactions.

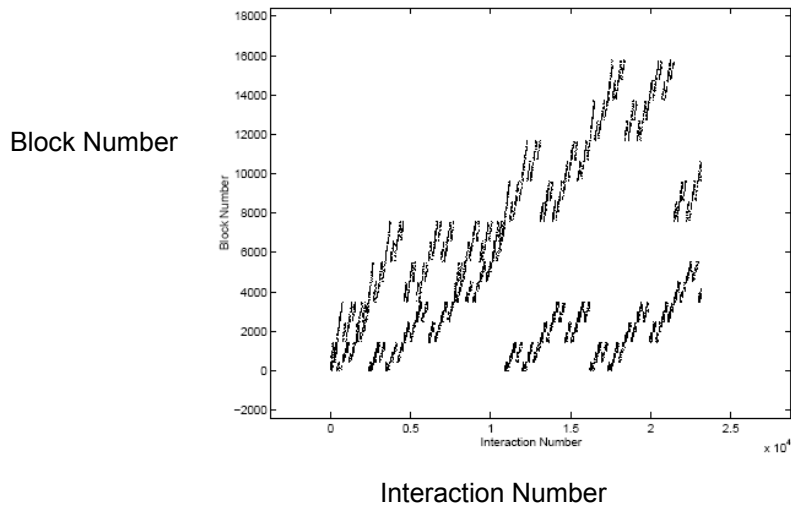
## Moldyn – base line program



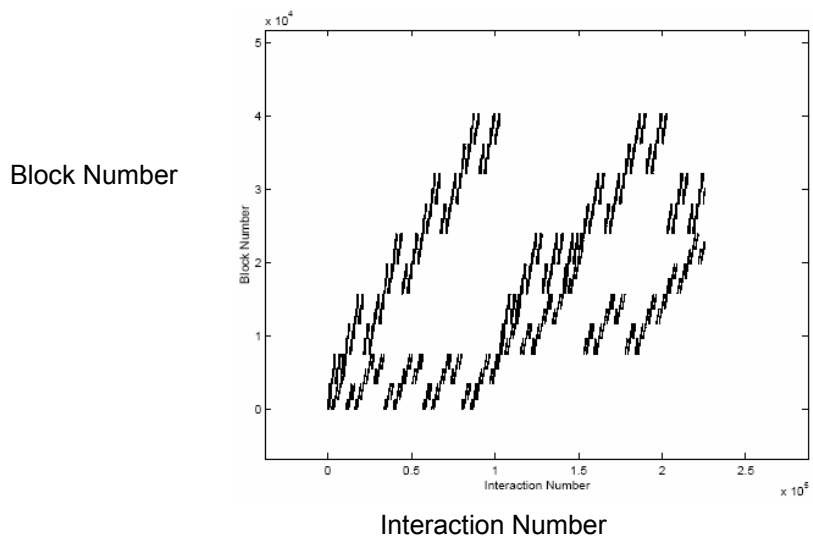
## Moldyn – Hilbert Curve



## Moldyn – Multi-level – first 10K

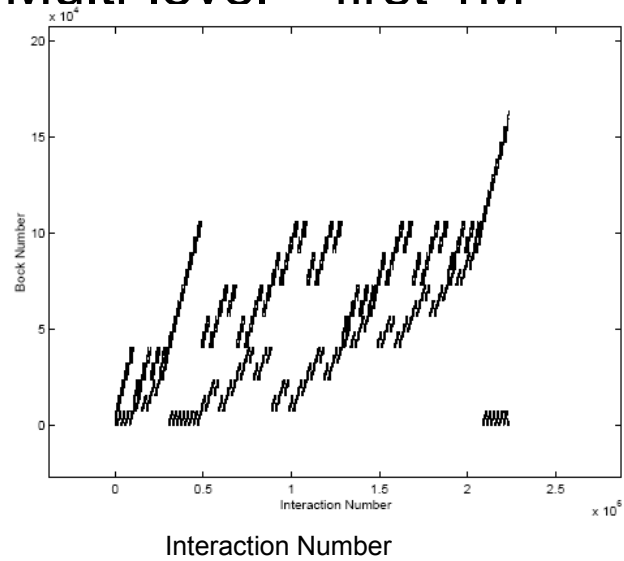


## Moldyn – Multi-level – first 100K



## Moldyn – Multi-level – first 1M

Block Number



## Moldyn – Various Reorderings

Data Reordering	Computation Reordering	L1 Cache Misses	L2 Cache Misses	TLB Misses	Cycles
RCM	None	0.96441	0.81847	0.49658	0.86650
First Touch	None	0.87487	0.76548	0.31928	0.79069
Hilbert	None	0.87978	0.78074	0.26397	0.80731
None	Hilbert	0.45053	0.12157	0.74006	0.37778
None	Blocking	0.30376	0.23557	0.19278	0.61910
First Touch	Hilbert	0.33735	0.14314	0.00806	0.38773
Hilbert	Hilbert	0.25816	0.10139	0.00624	0.26550

# Magi

- An application used by the U.S. Air Force to perform hydrodynamic computation.

- Description:

Read in the data for each of the particles.

**FOR**  $N$  time steps **DO**

**FOR** each particle  $i$  **DO**

        Create an interaction list for particle  $i$  containing neighbors within the sphere of influence.

**FOR** each particle  $j$  within this interaction list **DO**

            Update information for particle  $j$ .

Print the final results.

# Magi – Various Reorderings

Data Reordering	Computation Reordering	L1 Cache Misses	L2 Cache Misses	TLB Misses	Cycles
First Touch	First Touch	0.42959	0.27032	0.49173	0.56321
Hilbert	Hilbert	0.28621	0.11916	0.15704	0.43751
Hilbert/First Touch	Hilbert/First Touch	0.32670	0.11695	0.13513	0.43607

# CHAD

- A Parallel unstructured mesh application for simulating 3D fluid flows with chemical reactions and fuel sprays.
- Contains mostly dense vector operations.
- The results indicate that the original ordering is already good!

Node Reordering	Edge Reordering	L1 Cache Misses
original	lexicographic	0.962
Hilbert	original	1.28
Hilbert	lexicographic	0.978
Hilbert	Hilbert	1.03
RCM	lexicographic	0.972
RCM	Hilbert	1.01
random	original	1.61
random	lexicographic	1.92