ICS RESEARCH TECHNICAL TALK

DRAKE TETREAULT, ICS H197 FALL 2013

TOPIC: RESEARCH PAPER

- Title: Data Management for SSDs for Large-Scale Interactive Graphics Applications
- Authors: M. Gopi, Behzad Sajadi, Shan Jiang
- Published: 2011 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games

Data Management for SSDs for Large-Scale Interactive Graphics Applications

Behzad Sajadi *
Shan Jiang †
M. Gopi‡
Department of Computer Science
University of California, Irvine

Jae-Pil Heo §
Sung-Eui Yoon ¶
Korea Advanced Institute of Science and Technology

Figure 1: *A few snapshots of our interactive out-of-core walkthrough scene editing on The City Model (consisting of 110M triangles). The first image from left shows part of the original scene; the second one shows the scene after selecting an object (the house shown in green in the first image is chosen and shown in blue in the second image); the third image shows the scene after duplication of the same object and the last one after removing the original copy of the object.*

Abstract

Solid state drives (SSDs) are emerging as an alternative storage medium to HDDs. SSDs have performance characteristics (e.g., fast random reads) that are very different from those of HDDs. Because of the high performance of SSDs, there are increasingly more research efforts to redesign the established techniques that are optimized for HDDs, to work well with SSDs. In this paper we focus on computing cache-coherent layouts of large-scale models for SSDs. It has been demonstrated that cache-oblivious layouts perform well for various applications running on HDDs. However, computing cache-oblivious layouts for large-models is known to be very expensive. Also these layouts cannot be maintained efficiently for dynamically changing models. Utilizing the properties of SSDs we propose an efficient layout computation method that produces a page-based cache-aware layout for SSDs. We show that the performance of our layout can be maintained under dynamic changes on the model and is similar to the cache-oblivious layout optimized for static models. We demonstrate the benefits of our method for large-scale walkthrough scene editing and rendering, and collision detection.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—; I.3.8 [Computer Graphics]: Applications—;

Keywords: dynamic data layouts, cache-aware, cache-oblivious, cache-coherent layouts, out-of-core applications, walkthrough scene editing, walkthrough rendering, collision detection, solid state devices, solid state drives, flash drives, hard-disk drives.

*e-mail: bsajadi@uci.edu
†e-mail: sjiang4@ics.uci.edu
‡e-mail: gopi@ics.uci.edu
§e-mail: jaepil@jupiter.kaist.ac.kr
¶e-mail: sungeui@gmail.com

1 Introduction

Hard disk drives (HDDs) have been a very successful storage medium since the magnetic tape era. Thanks to their success, HDDs have inspired numerous research studies on various essential components of hardware and software including external data access models for out-of-core algorithms [Vitter 2001] and virtual memory schemes [Levy and Lipman 1982] for operating systems.

In addition to the general data management community, computer graphics and visualization researchers have also contributed tremendously to the management of graphics and visualization specific data using novel data structures, cache management techniques, and out-of-core algorithms for interactive rendering and many other graphics and visualization applications [Silva et al. 2002].

Recently, solid state drives (SSDs) (i.e. flash memory) are emerging as an alternative storage medium to HDDs. SSDs are widely adopted in mobile devices, laptops, and even desktop PCs, as the secondary storage. This phenomenon is mainly caused by their cheaper cost than DRAMs, and much faster access performance than HDDs [Agrawal et al. 2008].

SSDs have performance characteristics that are very different from those of HDDs. The random read performance of SSDs is much faster compared to HDDs. Moreover, the random read performance of SSDs is similar to their sequential read performance. Also, the sequential write performance of SSDs is similar to or even faster than their sequential and random read performance [Agrawal et al. 2008]. On the other hand, SSDs have performance that are relatively similar to DRAMs, except that SSDs are three times slower on average than DRAMs [1]. However, stored data in SSDs are non-volatile in contrast to DRAMs.

With the advent of SSDs, there are increasingly more research efforts to redesign the established techniques, that are optimized for

[1] http://www.bitmicro.com/press_resources_flash_ssd.php

OUTLINE

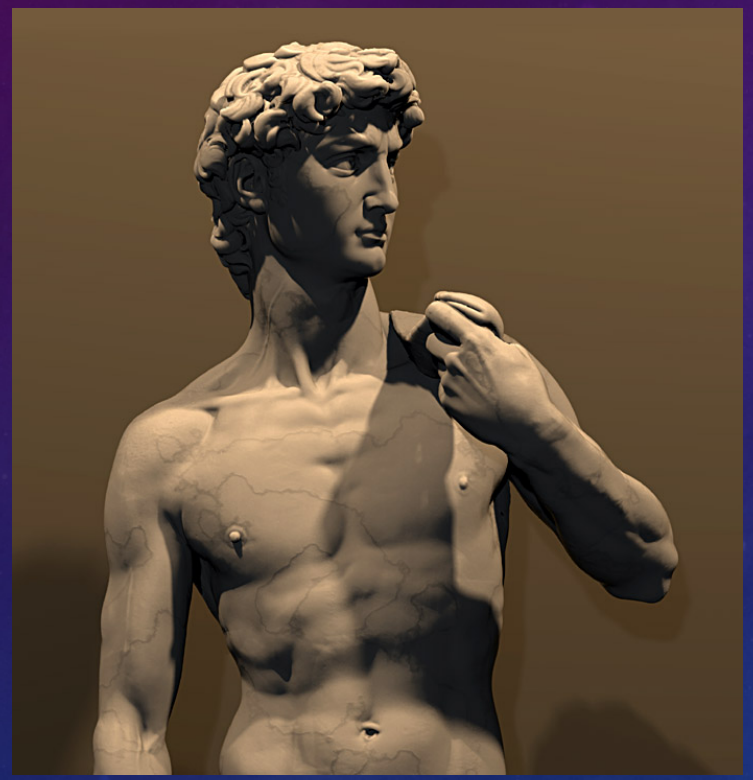
- Problem Statement
  - The Goal: Interactive Walkthrough of Massive Models
  - Desired Operations: Insert and Delete
- Preexisting Solutions Based on HDDs
  - Platter Drive Performance Characteristics
  - Cache Oblivious Algorithm
  - Limitations of Platter Drive Algorithm
- Proposed Solution Based on SSDs
  - SSD Performance Characteristics
  - Cache Aware Algorithm
  - Benefits of Cache Aware Layouts

MASSIVE MODEL RENDERING

- What is a massive model?
  - Any 3D model which does not fit in main memory.
  - Definition changes with increasing computer performance.
  - Requires “Out of Core” rendering algorithm.
- How many triangles?
  - 1 vertex = (x,y,z) * 4 bytes/coord = 12 bytes
  - 1 triangle = 3 vertices = 36 bytes
  - 1 GB / 36 bytes = 33.5M triangles (roughly)

The City Model: 110M triangles, 3.7 GB

MASSIVE MODEL EXAMPLE



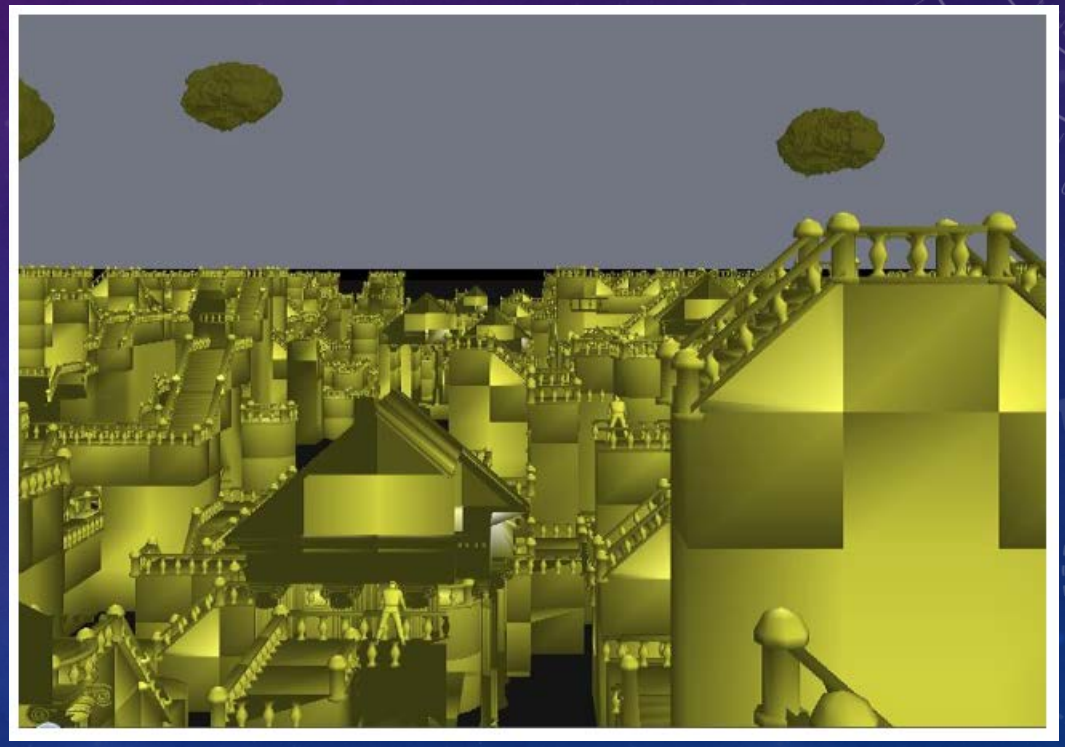
https://graphics.stanford.edu/papers/dmich-sig00/

- The dataset generated by the Digital Michelangelo Project's scanning of Michelangelo's *David* statue is a current example of a massive model.
- Raw data has over 2 billion triangles = 60 GB
- Takeaway: although computer power will increase, so will the complexity of models computers are asked to render

THE GOAL: WALKTHROUGH VIEWING

- Given such large models, the goal is to render them on a desktop computer at interactive frame rates.

- Interactive frame rates → 10 Hz minimum, although 60 Hz is preferred.



False color view of the city model

OPERATIONS: INSERT AND DELETE

- In addition to walkthrough viewing capabilities, it is also desirable to support real time editing of the models by users.

- Which operations will users want to perform?
  - Add geometry
  - Move geometry
  - Transform geometry
  - Remove geometry

- All of these can be reduced to a series of insert and delete triangle operations. A good solution should support both these operations.

http://www.nationofblue.com/attachment.php?attachmentid=9487&d=1327261970

OUTLINE

- Problem Statement
  - The Goal: Interactive Walkthrough of Massive Models
  - Desired Operations: Insert and Delete
- Preexisting Solutions Based on HDDs
  - Platter Drive Performance Characteristics
  - Cache Oblivious Algorithm
  - Limitations of Platter Drive Algorithm
- Proposed Solution Based on SSDs
  - SSD Performance Characteristics
  - Cache Aware Algorithm
  - Benefits of Cache Aware Layouts

EXISTING SOLUTIONS USE HARD DISK DRIVES

- Massive models by definition cannot fit entirely within main memory.
- At any time the majority of the model's vertex data will be stored out of core on persistent media.
- Before the arrival of SSDs, the only choice for large capacity storage was platter-based hard disk drives

http://www.hitechreview.com/gallery/western-digital-caviar-green-hdd/western-digital-caviar-green-hdd_5.jpg

PLATTER DRIVE PERFORMANCE CHARACTERISTICS

- Hard disk drives represent data as a one dimensional array of bytes.
- To retrieve data, a physical read head must be positioned over the correct platter location.
- The time taken for the read head to locate the start of a data section is called seek time. Modern HDD seek times average from 4 ms to 12 ms.
- For a 60 Hz framerate, each frame must render within 16 ms. This means that even a single data seek can consume the majority of a frame's available time.
- All algorithms designed for use on platter drives must account for their very poor random access performance.

CACHE OBLIVIOUS ALGORITHM

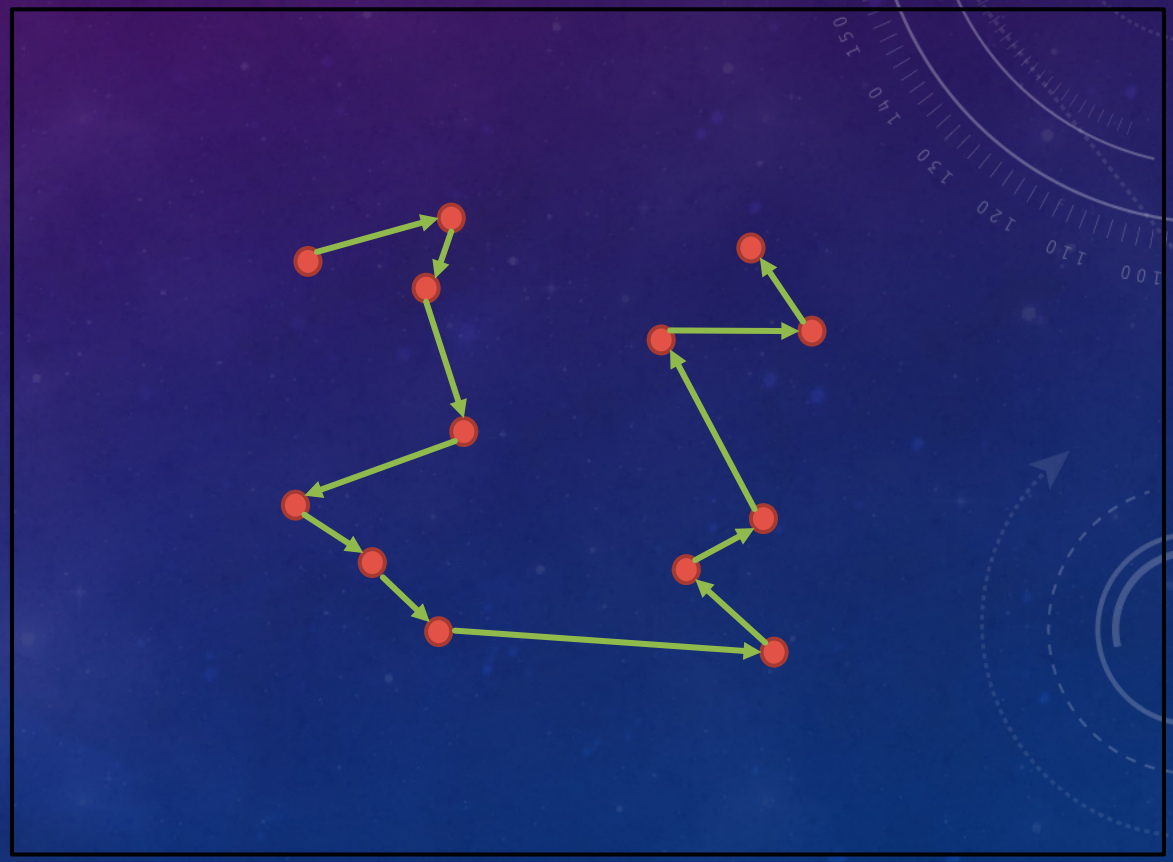
- Cache oblivious layouts attempt to globally optimize the on-disk order of geometric primitives to minimize the random seeks necessary for any arbitrary viewing position.
- Cache oblivious layouts are so named because the model data is organized sequentially without regard for the CPU's high performance data cache.
- They are computationally expensive to compute.
- Takes > 2 hours to compute the most efficient cache oblivious layout for the 110M triangle city model used in the paper.

CACHE OBLIVIOUS LAYOUT EXAMPLE

- Suppose you have the following 2D scene that you must organize into a 1D cache oblivious layout for storage on disk.

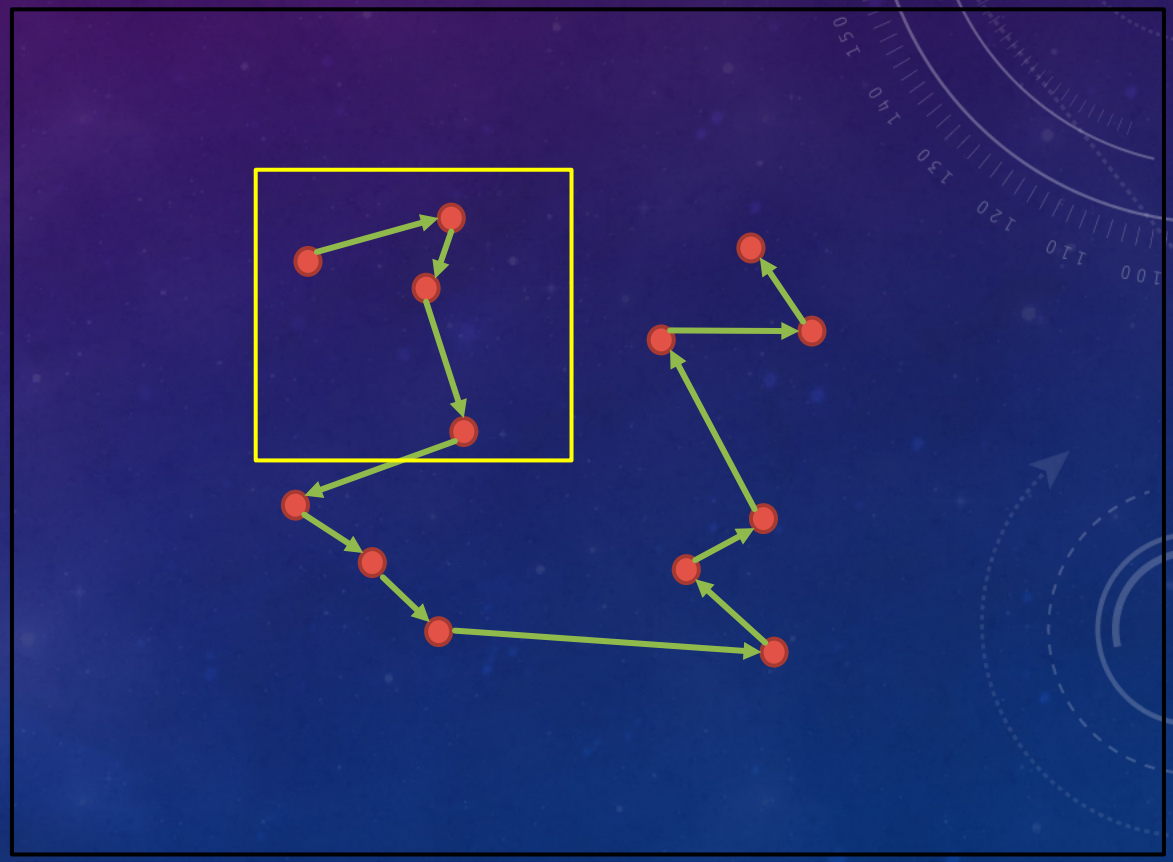
CACHE OBLIVIOUS LAYOUT EXAMPLE

- The cache oblivious layout will look something like this.



CACHE OBLIVIOUS LAYOUT EXAMPLE

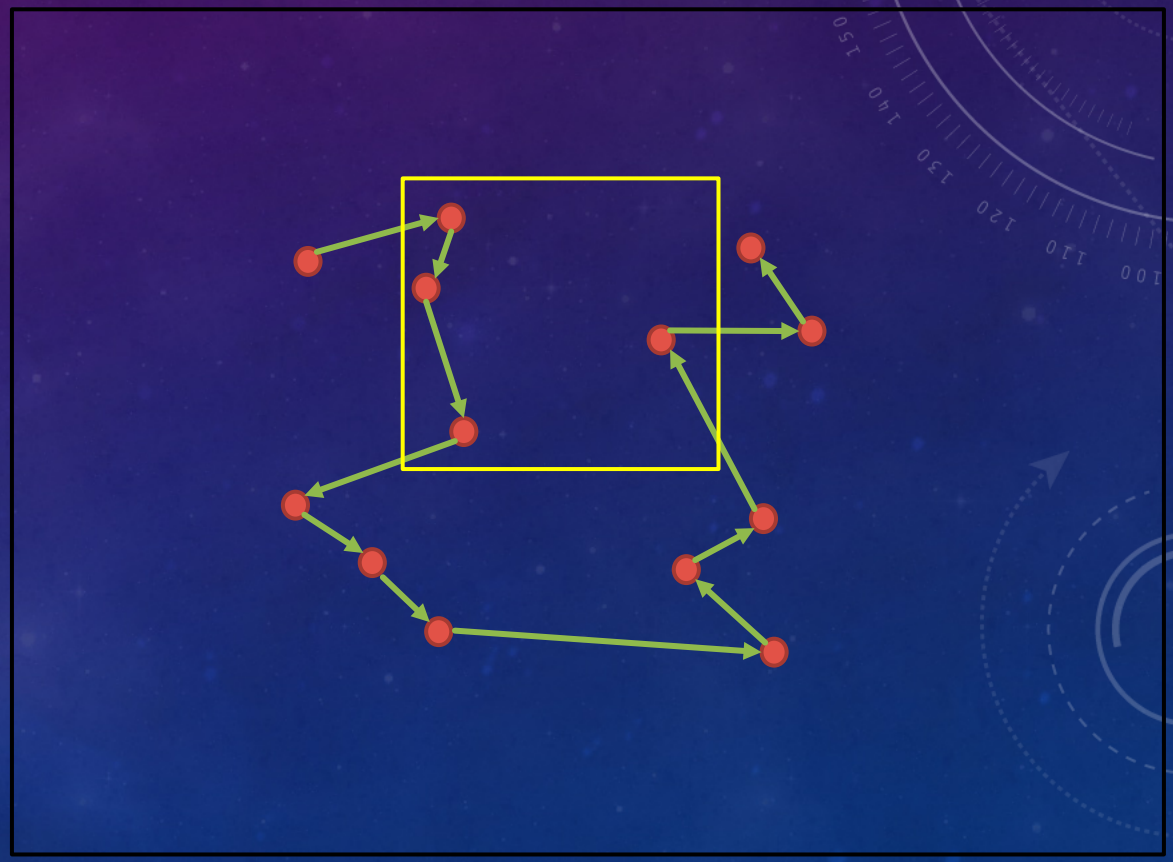
- There are many possible viewports.
- For a given viewport, ideally want to make no more than one data seek per frame.
- However, cache oblivious layouts cannot guarantee this.



1 Seek Required

CACHE OBLIVIOUS LAYOUT EXAMPLE

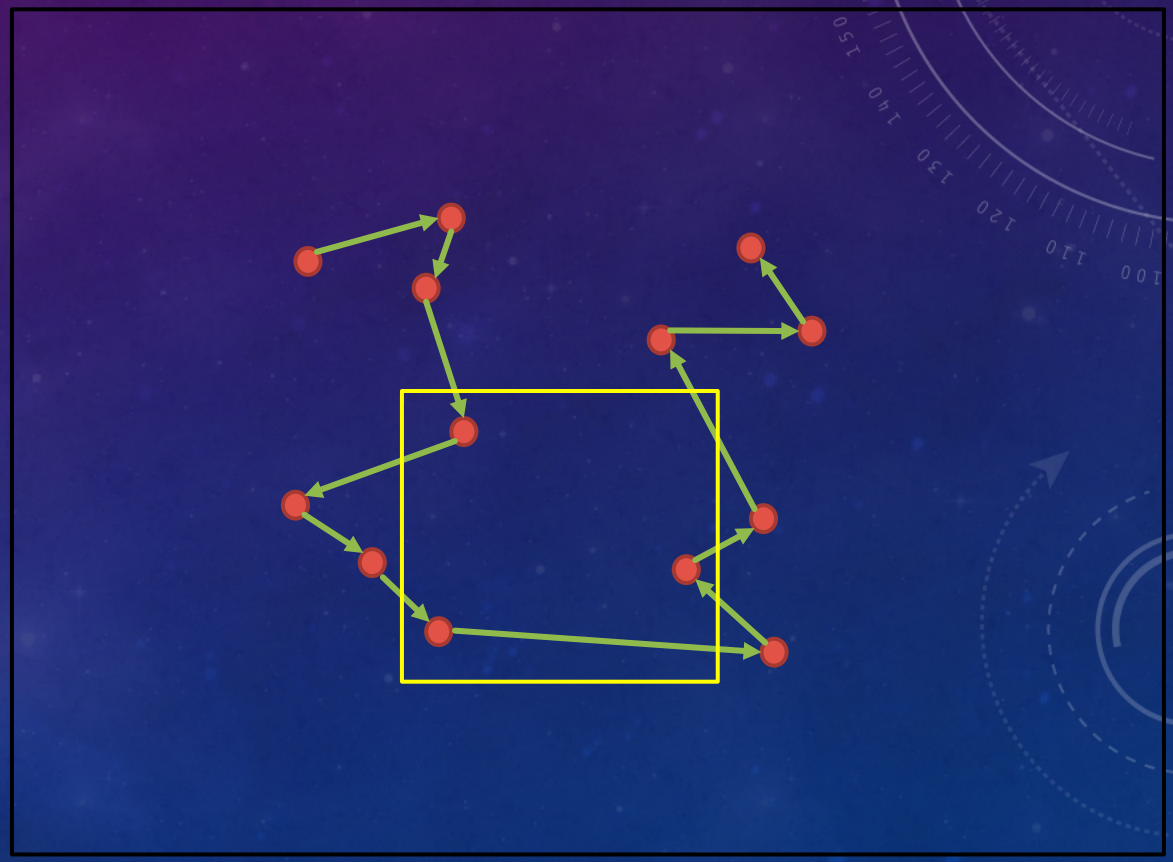
- There are many possible viewports.
- For a given viewport, ideally want only the objects inside the viewport to be loaded.
- However, cache oblivious layouts cannot guarantee this.



2 Seeks Required

CACHE OBLIVIOUS LAYOUT EXAMPLE

- There are many possible viewports.
- For a given viewport, ideally want only the objects inside the viewport to be loaded.
- However, cache oblivious layouts cannot guarantee this.



3 Seeks Required

PLATTER DRIVE ALGORITHM DRAWBACKS

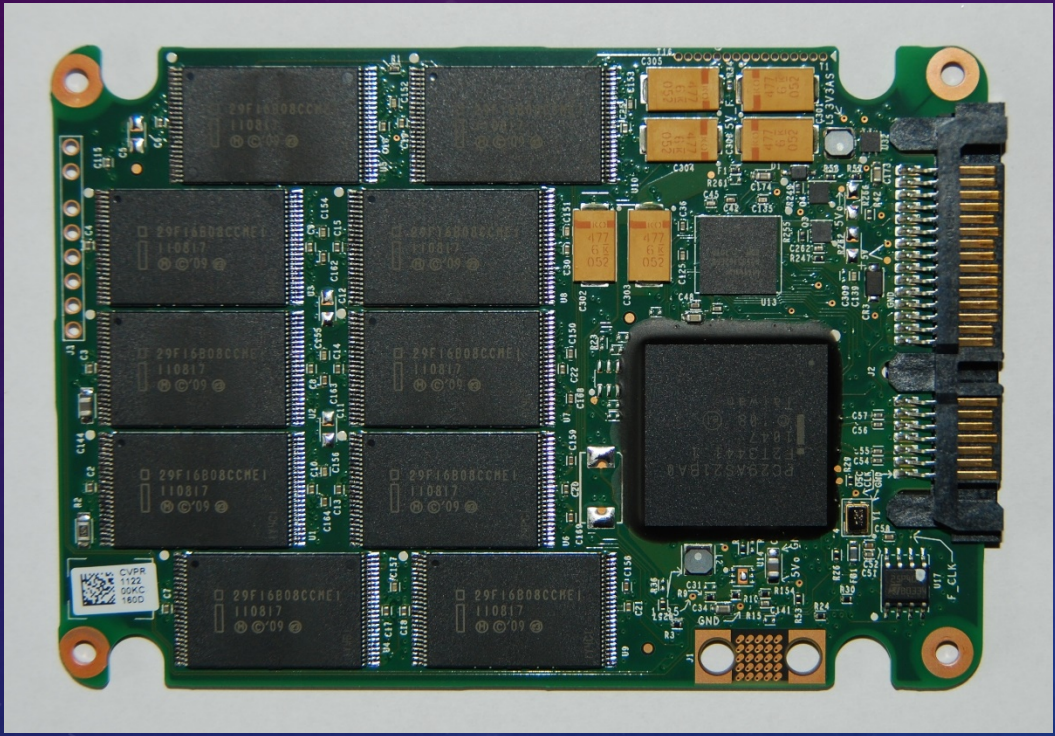
- Because algorithms intended for use on slow-random-access HDDs must make use of globally optimal cache oblivious layouts to minimize random seeks, they have inherent performance and feature limitations.

- Most significant limitation: cannot modify model at run time.
  - Because layout must be globally optimal, modification of even local parts would mandate recomputation of entire layout, but this can take hours.
  - This means that traditional massive model rendering solutions cannot efficiently support real time insert and delete operations.

OUTLINE

- Problem Statement
  - The Goal: Interactive Walkthrough of Massive Models
  - Desired Operations: Insert and Delete
- Preexisting Solutions Based on HDDs
  - Platter Drive Performance Characteristics
  - Cache Oblivious Algorithm
  - Limitations of Platter Drive Algorithm
- Proposed Solution Based on SSDs
  - SSD Performance Characteristics
  - Cache Aware Algorithm
  - Benefits of Cache Aware Layouts

SSD PERFORMANCE CHARACTERISTICS



Intel 530 SSD Interior – No Mechanical Parts

- Solid state drives (SSDs) are a new type of hard drive based on flash memory.
- They are electronic rather than mechanical, so they do not have a seek head which must be positioned before data can be read.
- Typical read times for SSDs are ~ 0.1 ms. Contrast with 4-12 ms seek times for HDDs.
- SSDs have no seek penalty. Random access is as fast as sequential access.

CACHE AWARE APPROACH

- Cache aware layouts are an alternative proposed by the authors to cache oblivious layouts, made possible by the very good random access performance of SSDs.

- Cache aware layouts group spatially coherent geometry into “pages” of data. These pages need not be sequentially laid out on the hard drive.

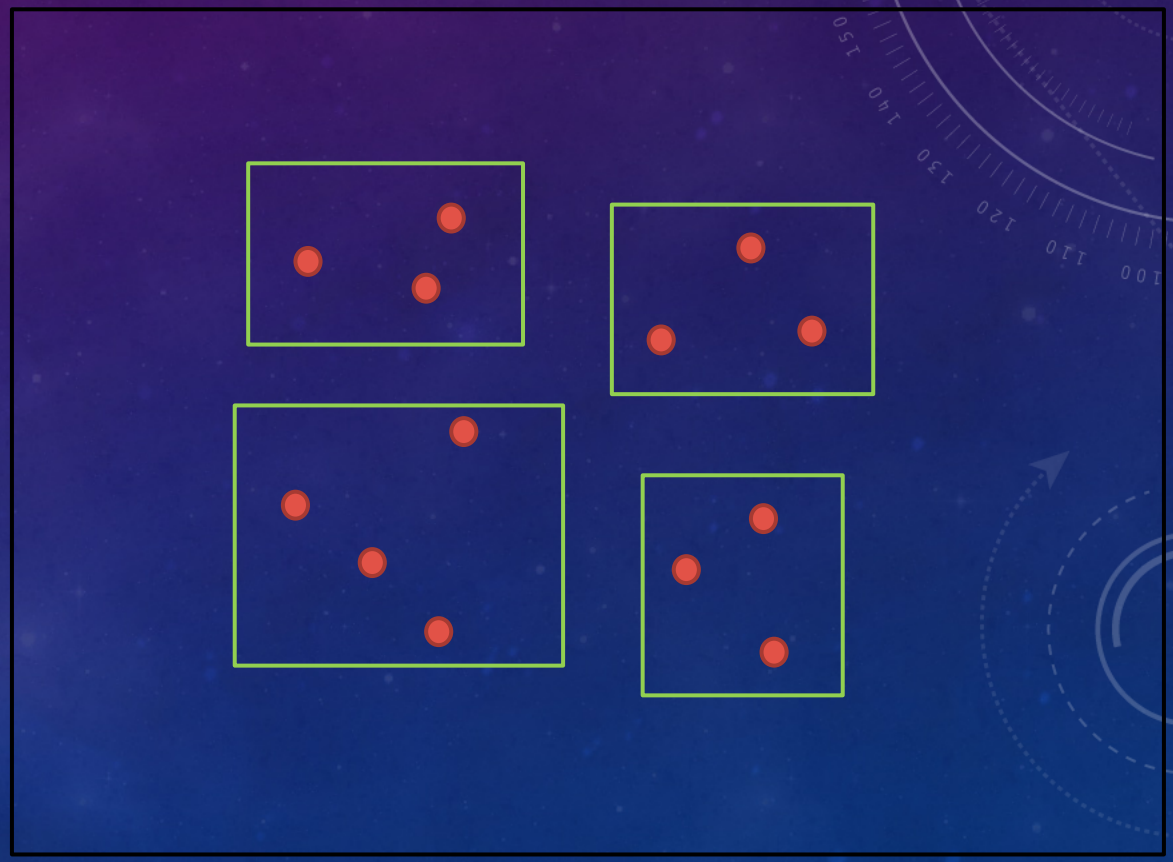
- Called “cache aware” because the pages are created in such a way as to fit into a CPU’s high speed data cache. In 2011 the authors used 4kb pages, but could use larger page sizes today due to larger CPU caches.

CACHE AWARE LAYOUT EXAMPLE

- Take the same 2D scene as before, but arrange it in a cache-aware layout instead of a cache oblivious layout.

CACHE AWARE LAYOUT EXAMPLE

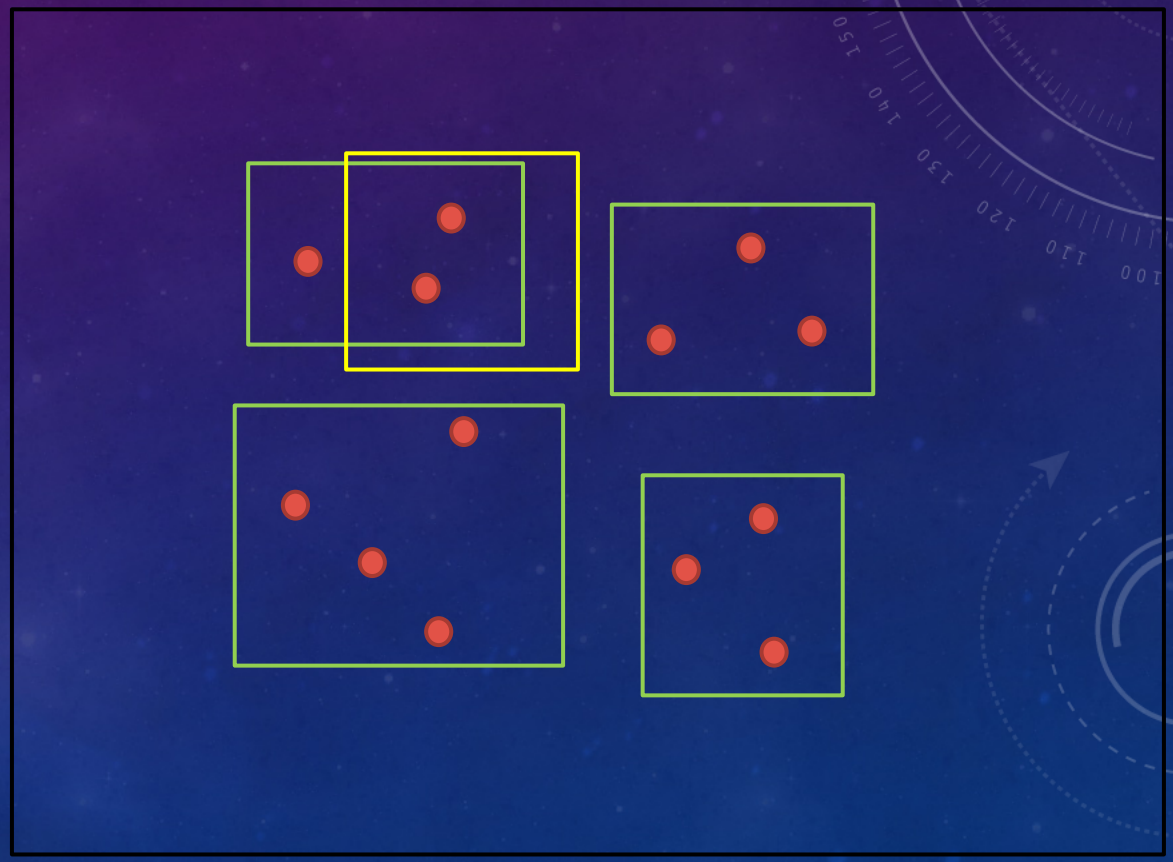
- The cache aware layout will look something like this.



CACHE AWARE LAYOUT EXAMPLE

- For any given viewport, intersect the viewport's bounding volume against the page's bounding volumes and load only those pages it intersects.

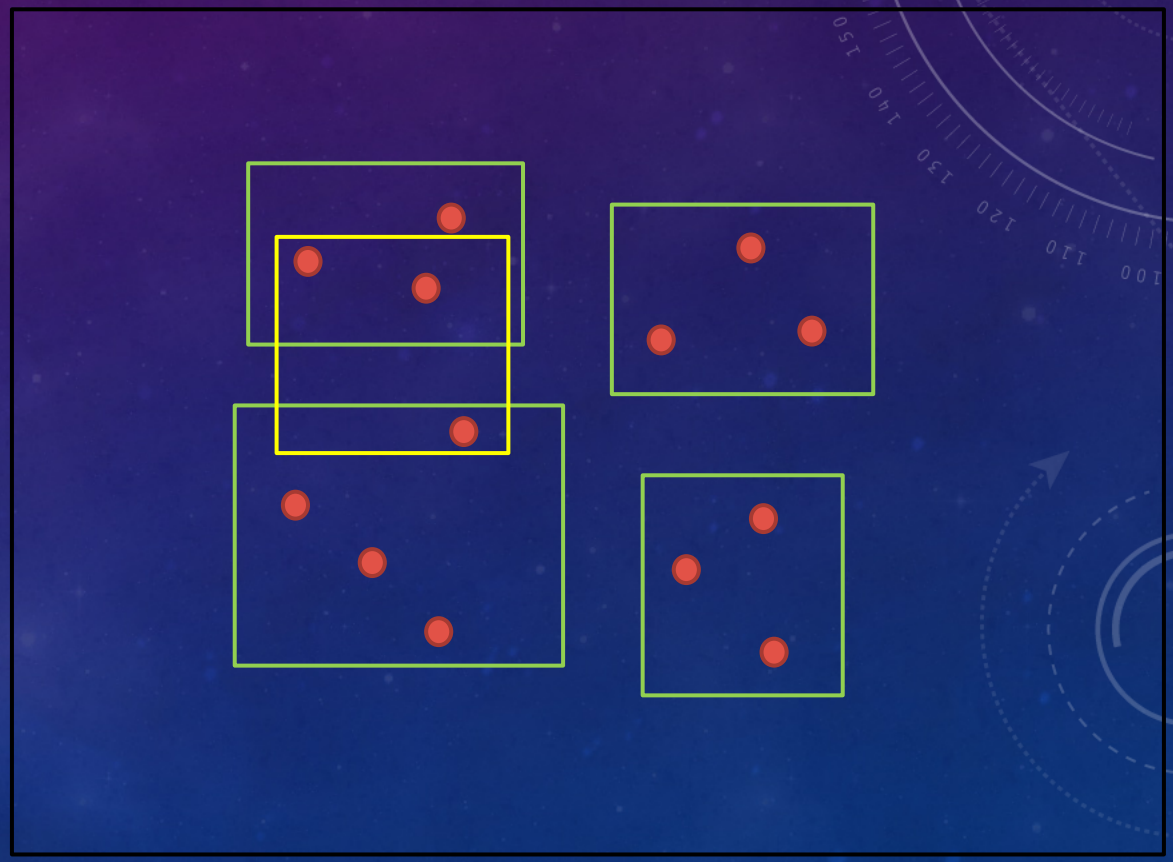
- Cache aware layouts guarantee an upper bound on the amount of data which is loaded, but not contained in the viewport, that is proportional to page size.



CACHE AWARE LAYOUT EXAMPLE

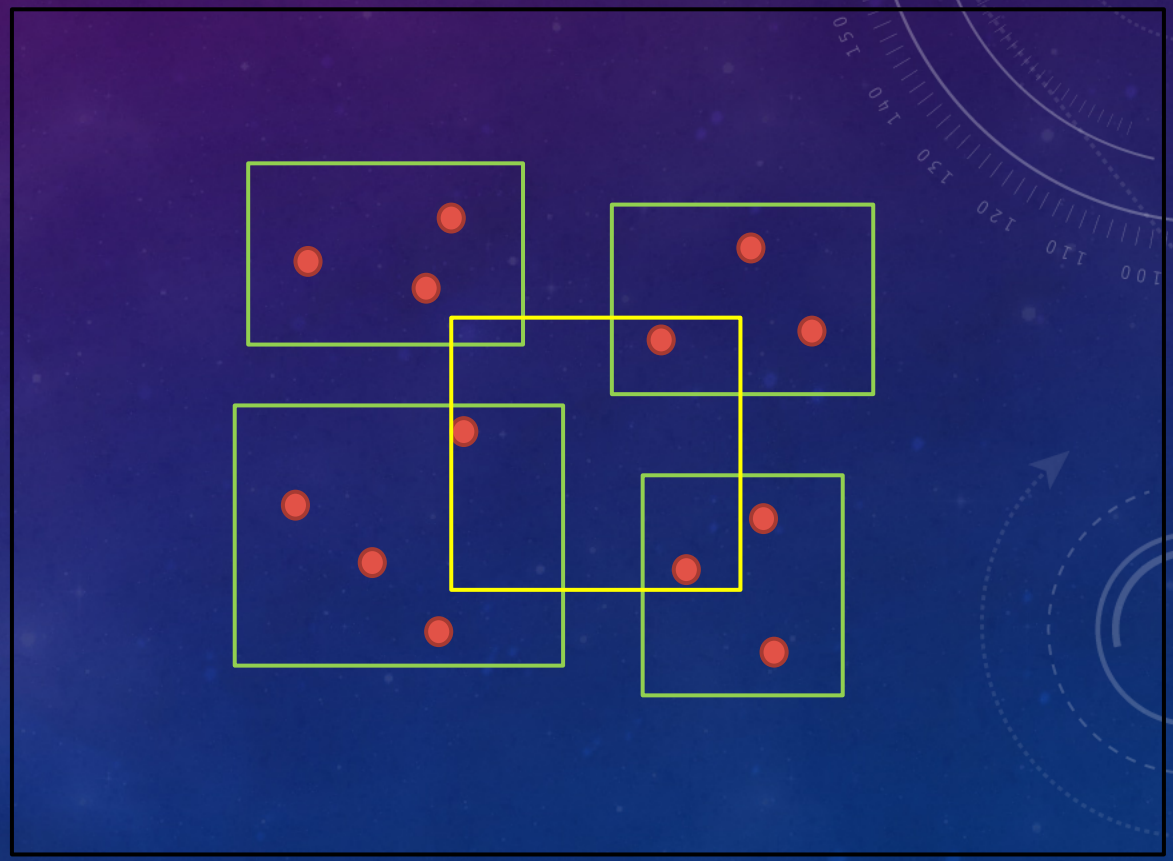
- For any given viewport, intersect the viewport's bounding volume against the page's bounding volumes and load only those pages it intersects.

- Cache aware layouts guarantee an upper bound on the amount of data which is loaded, but not contained in the viewport, that is proportional to page size.



CACHE AWARE LAYOUT EXAMPLE

- This is a worst case scenario, where the viewport intersects all four pages.
- Solution: if the number of pages to be loaded is higher than the number that can fit into main memory (for example, assume only 3 pages can be loaded), then sort the pages based on the distance from their centers to the viewport's center, and take those with the smallest distance.
- In this example, would not load the top left page because it is farthest from the viewport.



BENEFITS OF CACHE AWARE LAYOUTS

- Cache aware layouts have several benefits over cache oblivious layouts.
  1. They can be arranged on the SSD without regard to sequential order.

     All that needs to be kept in memory is a (small) table mapping page IDs to page file addresses.

BENEFITS OF CACHE AWARE LAYOUTS

- Cache aware layouts have several benefits over cache oblivious layouts.
  1. They can be arranged on the SSD without regard to sequential order.

     All that needs to be kept in memory is a (small) table mapping page IDs to page file addresses.

Page ID	Address
0	0x18ad33fe
1	0xf4792da4
2	0x0ad71b5c
…	…

BENEFITS OF CACHE AWARE LAYOUTS

- Cache aware layouts have several benefits over cache oblivious layouts.
  2. Page bounding boxes can be arranged in hierarchical data structures that support efficient viewport intersection queries, such as octrees and k-d trees.

     Such a hierarchy can be used for visibility tests such as frustum culling as well as dynamic level of detail scaling based on a page's distance to camera and screen space error metrics.

     A bounding box takes only 24 bytes to represent, so all pages' bounding boxes can fit into main memory at one time.

(x,y,z) * (upper left, upper right) * 4 bytes/number = 24 bytes/bounding box

BENEFITS OF CACHE AWARE LAYOUTS

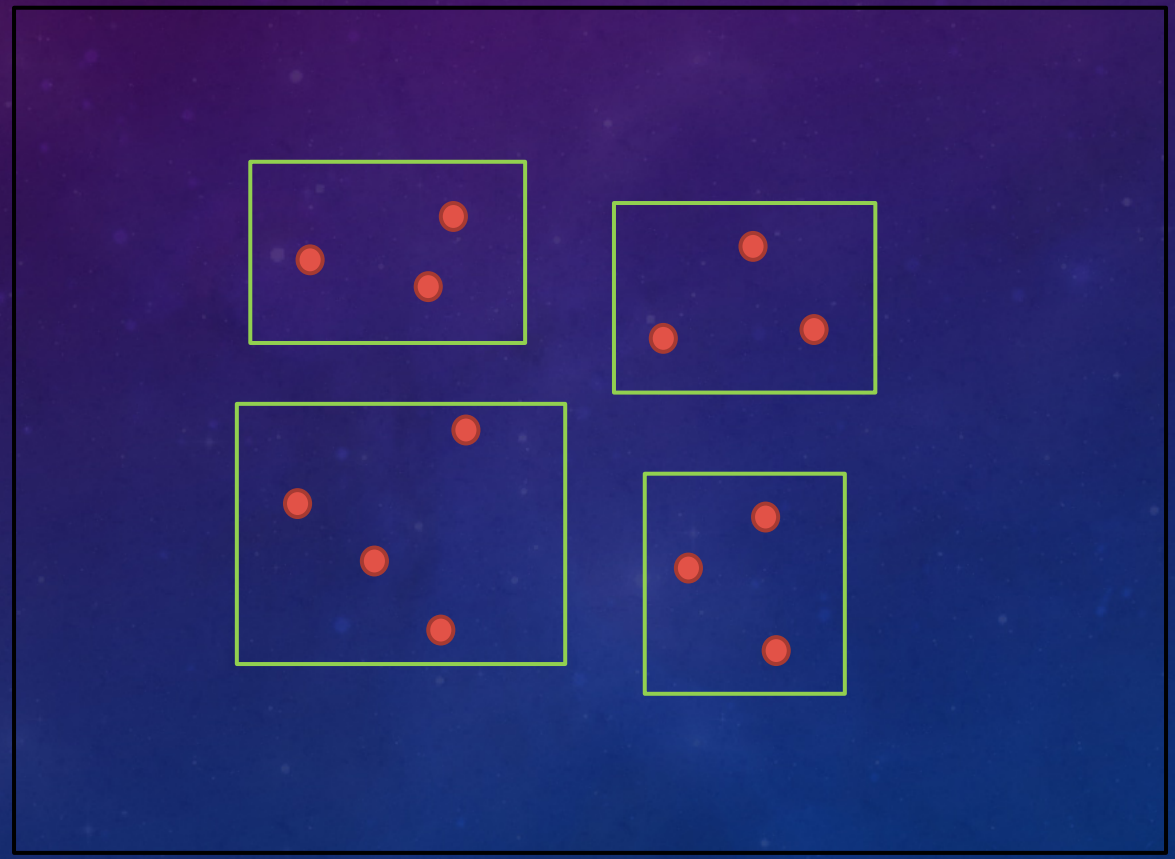
- Cache aware layouts have several benefits over cache oblivious layouts.

3. They are very easy to compute. For the 110M triangle city model used in the paper, it took 10 minutes to compute the initial full cache aware layout (contrast with 2+ hours to compute the cache oblivious layout).

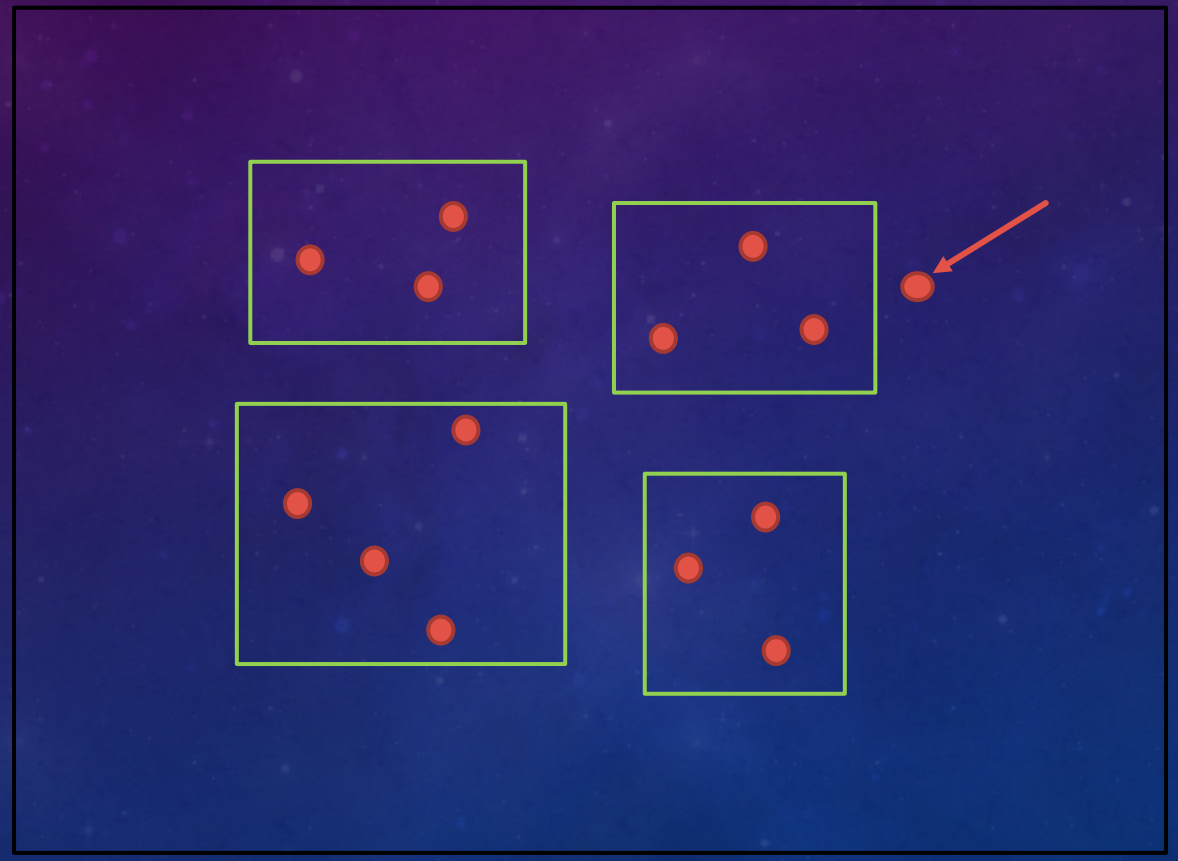
   However, that 10 minutes was the time taken to compute every single page's optimum composition. Individual pages can be refined using constant-time procedures.

   This is the greatest advantage of cache aware layout strategies over cache oblivious layout strategies. Because cache aware layouts are only locally optimal, the entire scene layout does not have to be recomputed for a single change, unlike cache oblivious layout strategies.

LOCAL RECOMPUTATION EXAMPLE

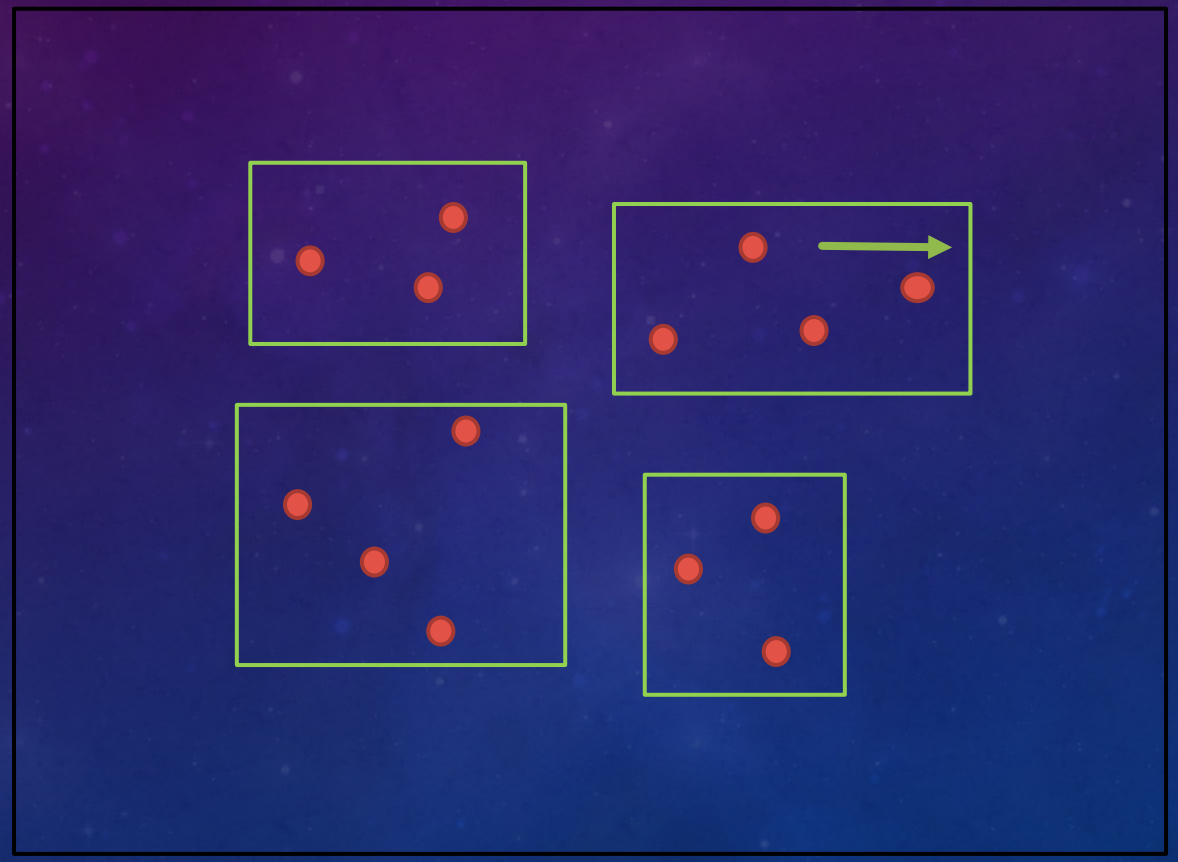


LOCAL RECOMPUTATION EXAMPLE



Local change…

LOCAL RECOMPUTATION EXAMPLE



… leads to local update.

BENEFITS OF CACHE AWARE LAYOUTS

- Cache aware layouts have several benefits over cache oblivious layouts.
    4. Because pages can be locally updated in constant time, cache aware layouts support interactive insertion and deletion operations like the one shown previously.

REVIEW

- Goal
  - To implement interactive rendering of massive models that don’t fit into memory, while supporting triangle insertion and deletion operations.
- Old Solution Based on HDDs
  - Traditional hard disk drive algorithms rely on cache oblivious layouts which are slow to compute, have no guaranteed upper bound on seek counts, and cannot be modified at run time.
- New SSD Based Solution
  - Uses the cache aware layout proposed by the authors of the paper to take advantage of solid state drives’ high random seek performance to enable fast preprocessing and support runtime modification of the massive model.

THANK YOU

QUESTIONS?