**Sony Computer Entertainment Europe
Research & Development Division**

Pitfalls of Object Oriented Programming

Tony Albrecht – Technical Consultant
Developer Services

What I will be covering

- A quick look at Object Oriented (OO) programming

- A common example

- Optimisation of that example

- Summary

Object Oriented (OO) Programming

- What is OO programming?
  - a programming paradigm that uses "objects" – data structures consisting of datafields and methods together with their interactions – to design applications and computer programs.
    (Wikipedia)

- Includes features such as
  - Data abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

What's OOP for?

- OO programming allows you to think about problems in terms of objects and their interactions.

- Each object is (ideally) self contained
    - Contains its own code and data.
    - Defines an interface to its code and data.

- Each object can be perceived as a 'black box'.

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

Objects

- ## If objects are self contained then they can be
  - – Reused.
  - – Maintained without side effects.
  - – Used without understanding internal implementation/representation.
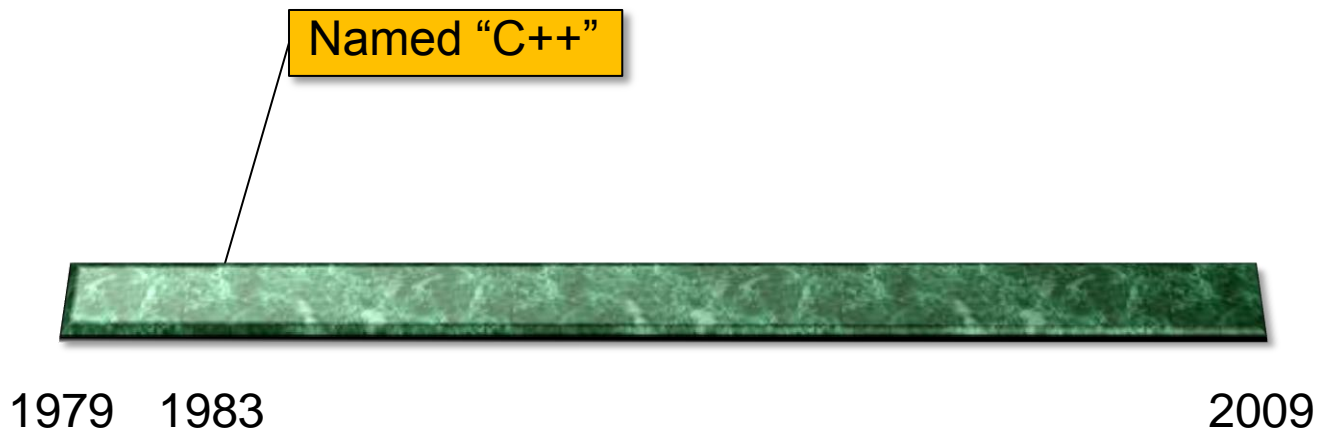
- ## This is good, yes?

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Are Objects Good?

- Well, yes

- And no.

- First some history.

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

C++ development started

1979

2009

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

Named "C++"

1979   1983   2009

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

First Commercial release

1979   1985   2009

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

Release of v2.0

1979

1989

2009

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

Release of

Added
• multiple inheritance,
• abstract classes,
• static member functions,
• const member functions
• protected members.

1979   1989   2009

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

Standardised

1979

1998

2009

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

Updated

1979   2003   2009

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

A Brief History of C++

C++0x

1979   2009   ?

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

So what has changed since 1979?

- Many more features have been added to C++
- CPUs have become much faster.
- Transition to multiple cores
- Memory has become faster.

- 16-bit microprocessor
- 16K RAM
- 13" color monitor (24 lines of 32 chrs.)
- 26K ROM operating system (includes 14K BASIC)
- Sound - 3 tones, 5 octaves
- 16 colors: 192 x 256 res.
- Large TI library of ROM programs available.

only $1150

FINALLY TEXAS INSTRUMENTS TI-99/4 Home Computer

Includes 13" Color Monitor!

Many Peripherals. Coming soon!

Over 1000 software tapes, books, disks on display. Come in and brouse.

http://www.vintagecomputing.com

CPU performance

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Computer architecture: a quantitative approach
By John L. Hennessy, David A. Patterson, Andrea C. Arpaci-Dusseau

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

CPU/Memory performance

Computer architecture: a quantitative approach
By John L. Hennessy, David A. Patterson, Andrea C. Arpaci-Dusseau

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

What has changed since 1979?

- One of the biggest changes is that memory access speeds are far slower (relatively)
  - 1980: RAM latency ~ 1 cycle
  - 2009: RAM latency ~ 400+ cycles

- What can *you* do in 400 cycles?

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

What has this to do with OO?

- OO classes encapsulate code and data.
- So, an instantiated object will generally contain all data associated with it.

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

My Claim

- With modern HW (particularly consoles), excessive encapsulation is BAD.

- Data flow should be fundamental to your design (Data Oriented Design)

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Consider a <u>simple</u> OO Scene Tree

- Base Object class
  - Contains general data
- Node
  - Container class
- Modifier
  - Updates transforms
- Drawable/Cube
  - Renders objects

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
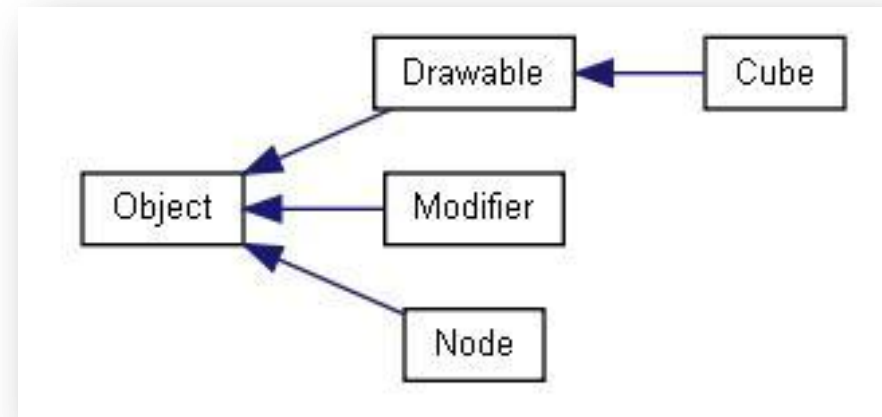
Object

- Each object
  - Maintains bounding sphere for culling
  - Has transform (local and world)
  - Dirty flag (optimisation)
  - Pointer to Parent

```
class Object
{
    // <methods removed for clarity>
    Matrix4 m_Transform;
    Matrix4 m_WorldTransform;
    BoundingSphere m_BoundingSphere;
    BoundingSphere m_WorldBoundingSphere;
    char* m_Name;
    bool m_Dirty;
    Object* m_Parent;
};
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Objects

Class Definition

Each square is 4 bytes

Memory Layout

```
class Object
{
    // <methods removed for clarity>
    Matrix4 m_Transform;
    Matrix4 m_WorldTransform;
    BoundingSphere m_BoundingSphere;
    BoundingSphere m_WorldBoundingSphere;
    char* m_Name;
    bool m_Dirty;
    Object* m_Parent;
};
```

vtbl			
Matrix4			
Matrix4			
Vector3			radius
Vector3			radius
char*	bool	Object*	

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
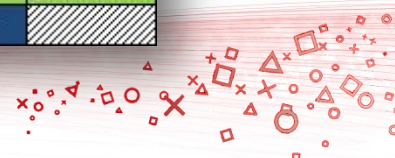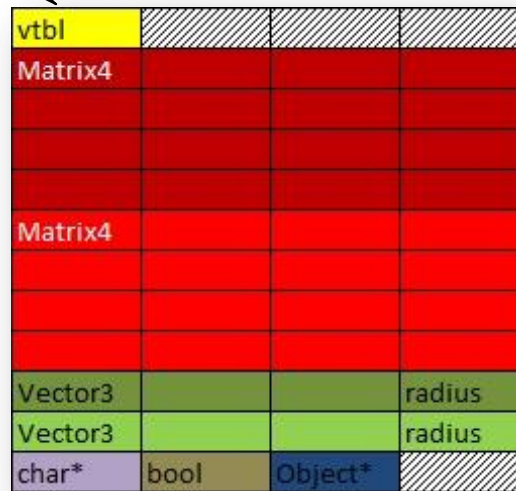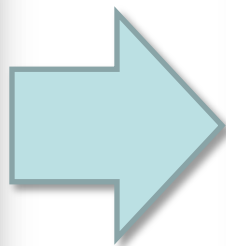
Nodes

- Each Node is an object, plus
  - Has a container of other objects
  - Has a visibility flag.

```cpp
class Node : public Object
{
    // methods removed for clarity
    std::vector<Object*> m_Objects;
    bool m_IsVisible;
};
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Nodes

Class Definition

```cpp
class Node : public Object
{
    // methods removed for clarity
    std::vector<Object*> m_Objects;
    bool m_IsVisible;
};
```

Memory Layout

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
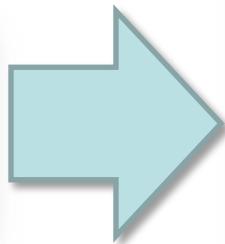
Consider the following code…

- Update the world transform and world space bounding sphere for each object.

```cpp
const BoundingSphere& Node::GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    m_WorldTransform = parentTransform*m_Transform;

    for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
        itr!= m_Objects.end();
        ++itr)
    {
        m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
    }
    return m_WorldBoundingSphere;
}
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Consider the following code…

- Leaf nodes (objects) return transformed bounding spheres

```
virtual const BoundingSphere& GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```
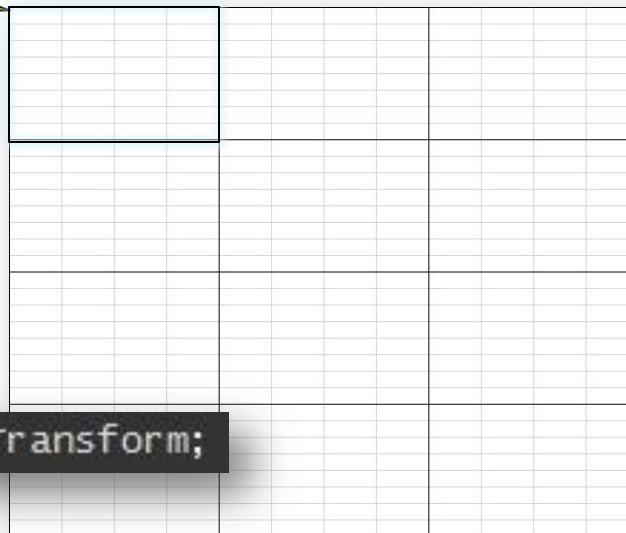
LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Consider the following code…

- Leaf nodes (objects) return transformed bounding spheres

> What's wrong with this code?

```
virtual const BoundingSphere& GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Consider the following code…

- Leaf nodes (objects) return transformed bounding

> If m_Dirty=false then we get branch misprediction which costs 23 or 24 cycles.

```
virtual const BoundingSphere& GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```

Consider the following code…

- Leaf nodes (objects) return transformed bounding

Calculation of the world bounding sphere takes only 12 cycles.

```
virtual const BoundingSphere& GetWorldBoundingSphere(          Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```

Consider the following code…

- Leaf nodes (objects) return transformed bounding

So using a dirty flag here is actually slower than not using one (in the case where it is false)

```
virtual const BoundingSphere& GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```

Lets illustrate cache usage

Main Memory

Each cache line is 128 bytes

L2 Cache

```
m_WorldTransform = parentTransform*m_Transform;
```

Cache usage

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
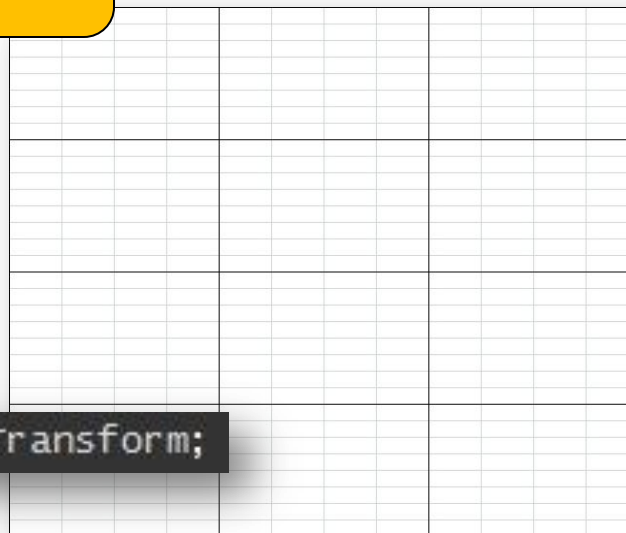
Cache usage

Main Memory

L2 Cache

Load m_Transform into cache

```
m_WorldTransform = parentTransform*m_Transform;
```

Cache usage

Main Memory
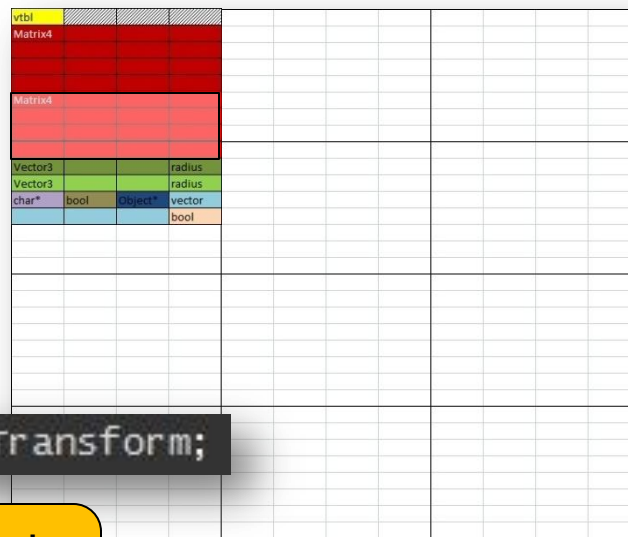
L2 Cache


```
m_WorldTransform = parentTransform*m_Transform;
```

m_WorldTransform is stored via cache (write-back)
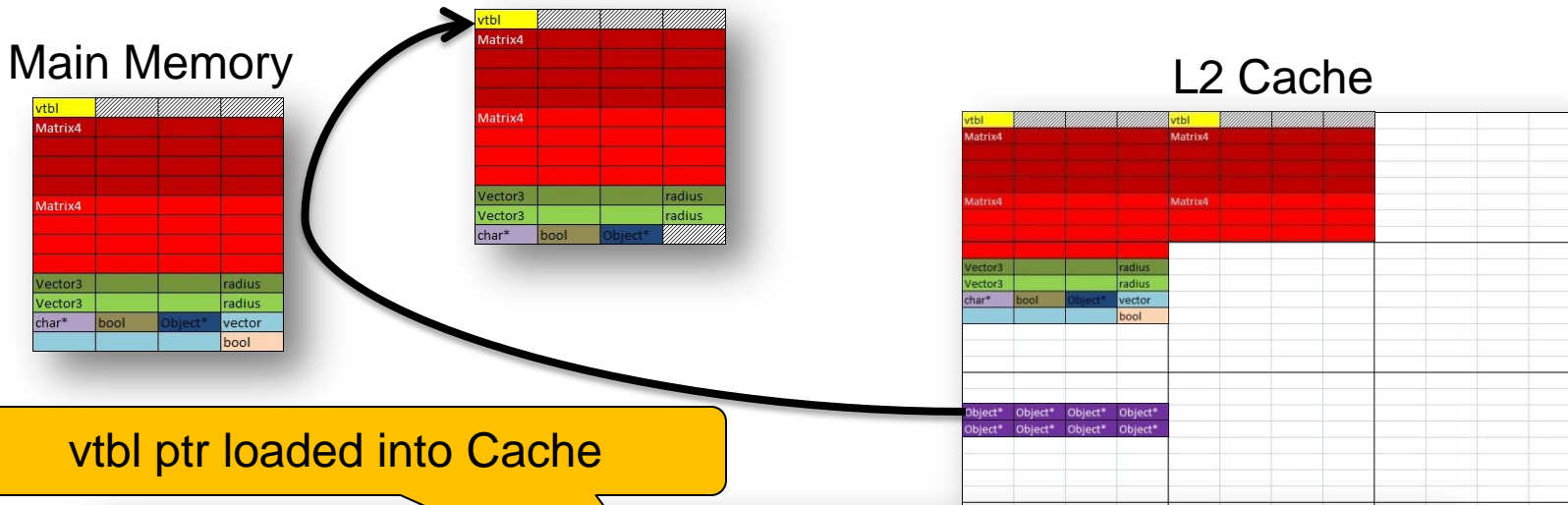
Cache usage

Main Memory
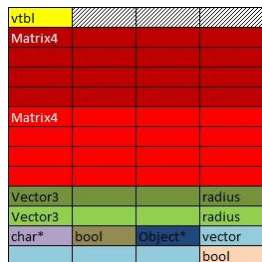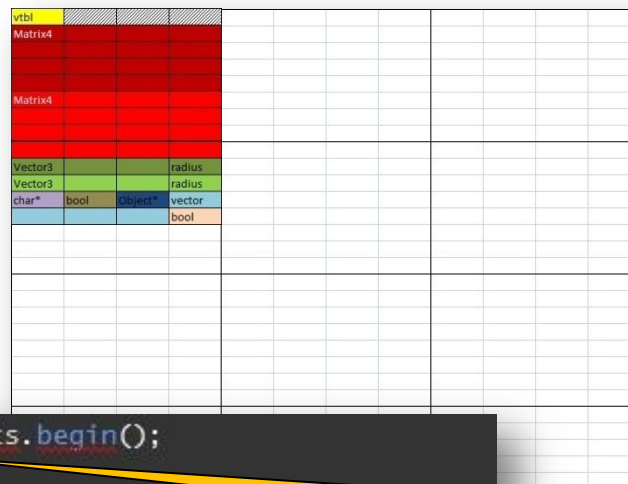
L2 Cache


```
for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSp
}
```

Next it loads m_Objects

Cache usage

Main Memory
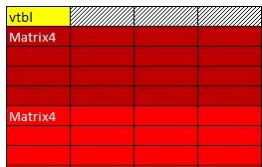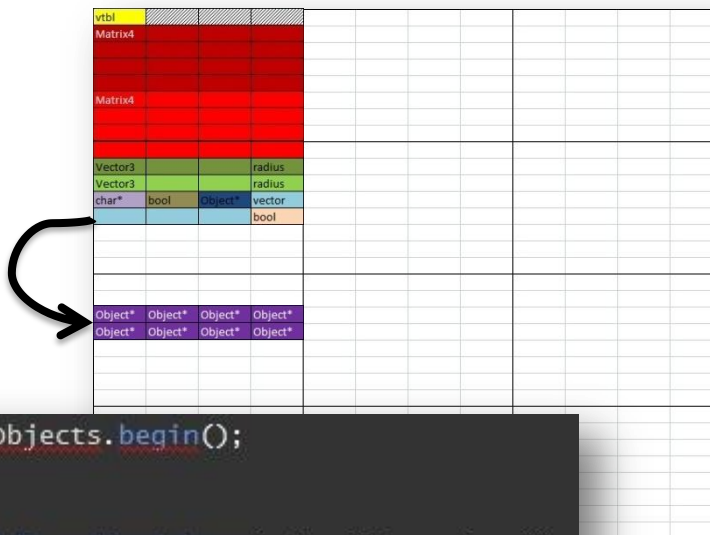
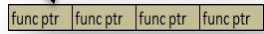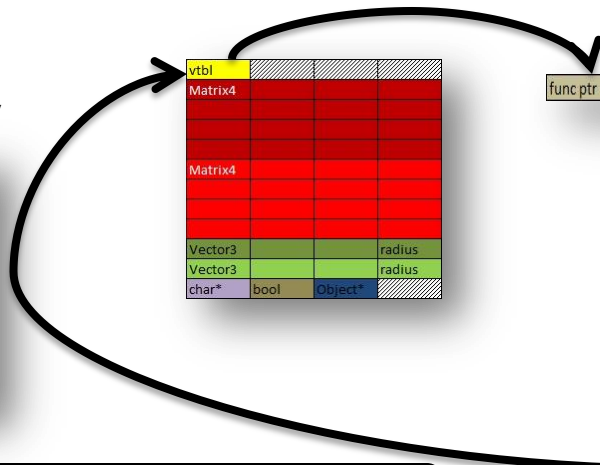L2 Cache

vtbl ptr loaded into Cache

```
for(std::vector<Object*>::iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
}
```
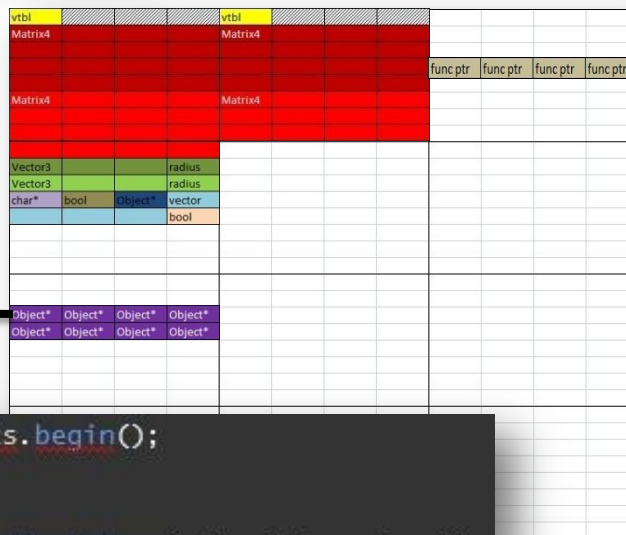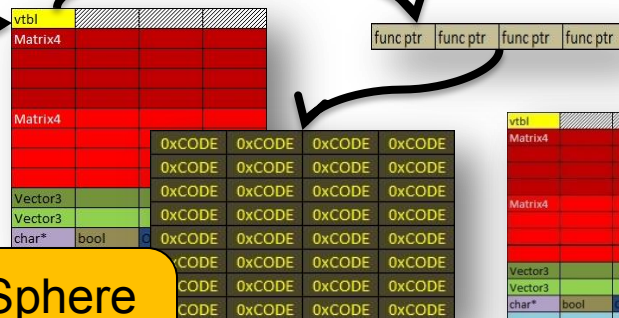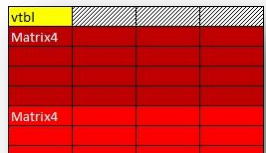
Cache usage

Main Memory

L2 Cache

Look up virtual function

```
for(std::vector<Object*>::iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
}
```

Cache usage

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Cache usage

Main Memory

L2 Cache

New code checks dirty flag then
sets world bounding sphere

```
const BoundingSphere& GetWorldBoundingSphere(const Matrix4& parentTransform)
{
    if(m_Dirty)
        m_WorldBoundingSphere = m_BoundingSphere.Transform(parentTransform);
    return m_WorldBoundingSphere;
}
```

Cache usage

Main Memory

L2 Cache

Node's World Bounding Sphere is then Expanded

```
for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
}
```
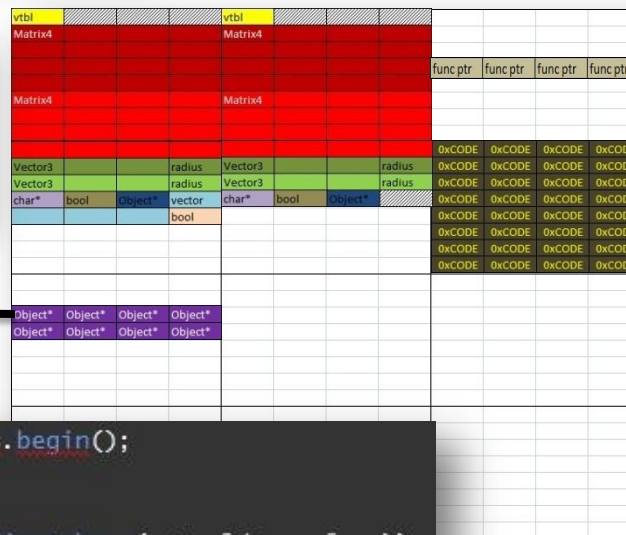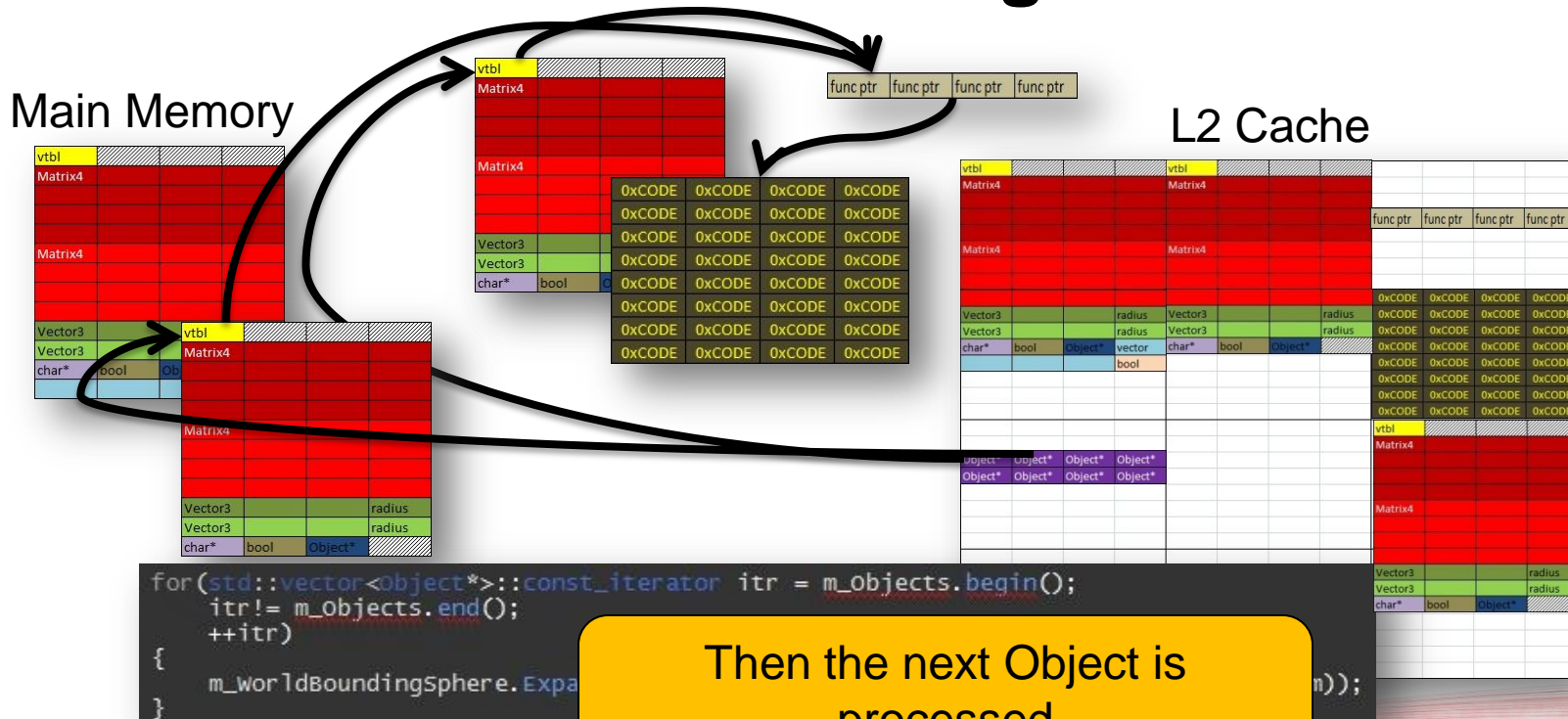
Cache usage

Main Memory

L2 Cache

```
for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.Expa...                  n));
}
```

Then the next Object is processed

Cache usage
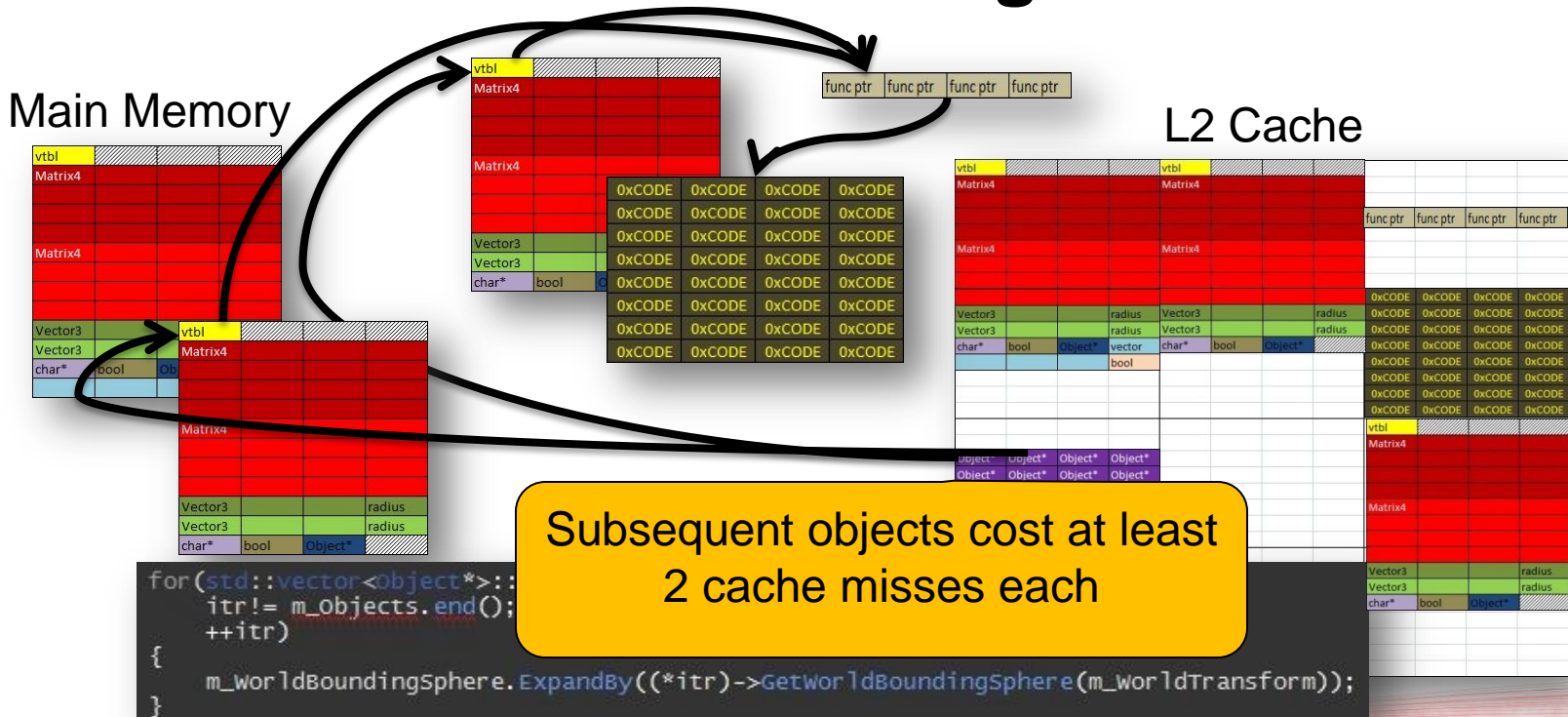
Main Memory

L2 Cache

```
for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.Expa...                  m));
}
```

First object costs at least 7 cache misses

Cache usage

Main Memory

L2 Cache

Subsequent objects cost at least
2 cache misses each

```
for(std::vector<Object*>::
    itr!= m_Objects.end();
    ++itr)
{
    m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
}
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
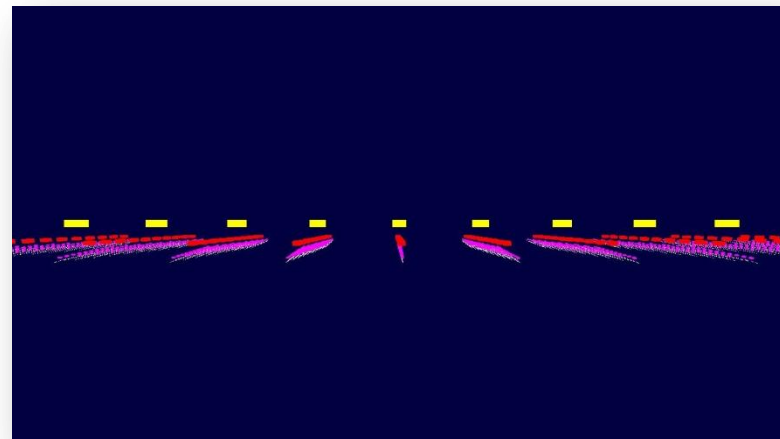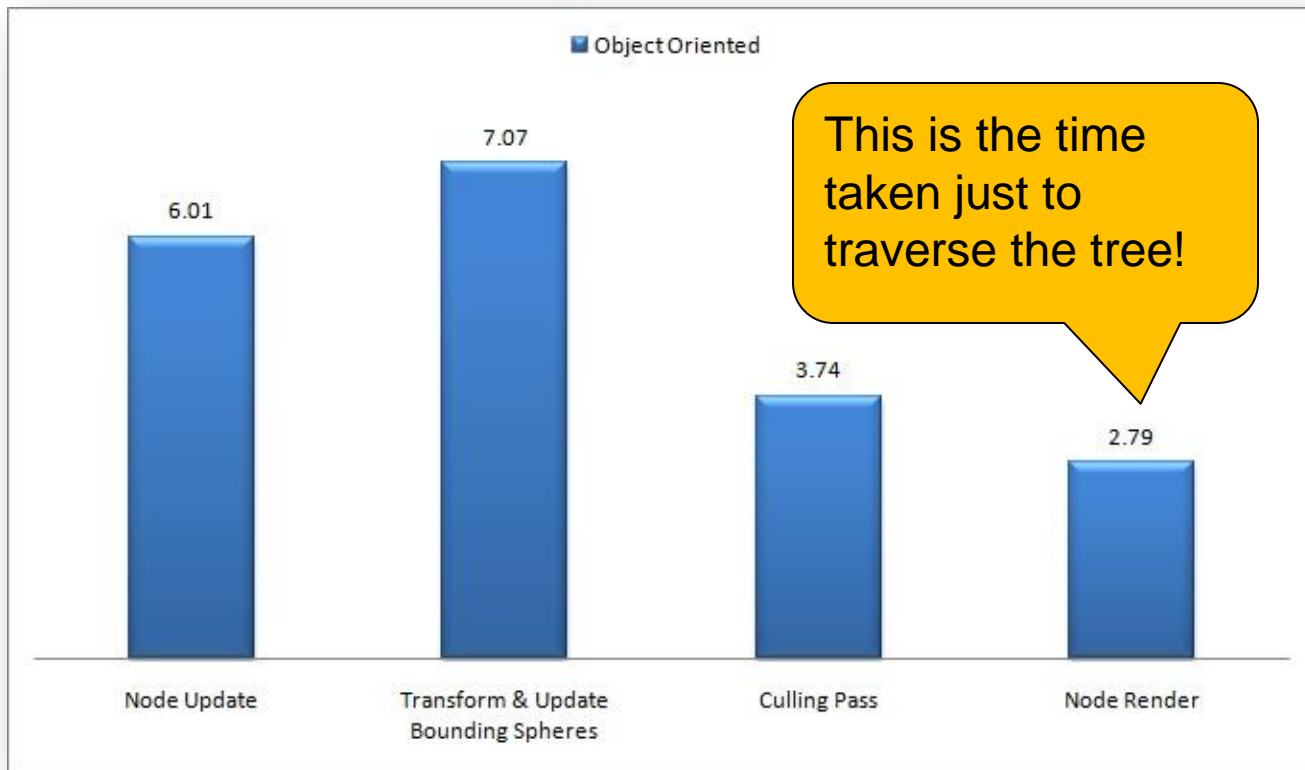
The Test

- 11,111 nodes/objects in a tree 5 levels deep
- Every node being transformed
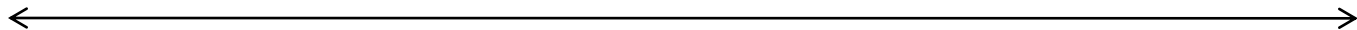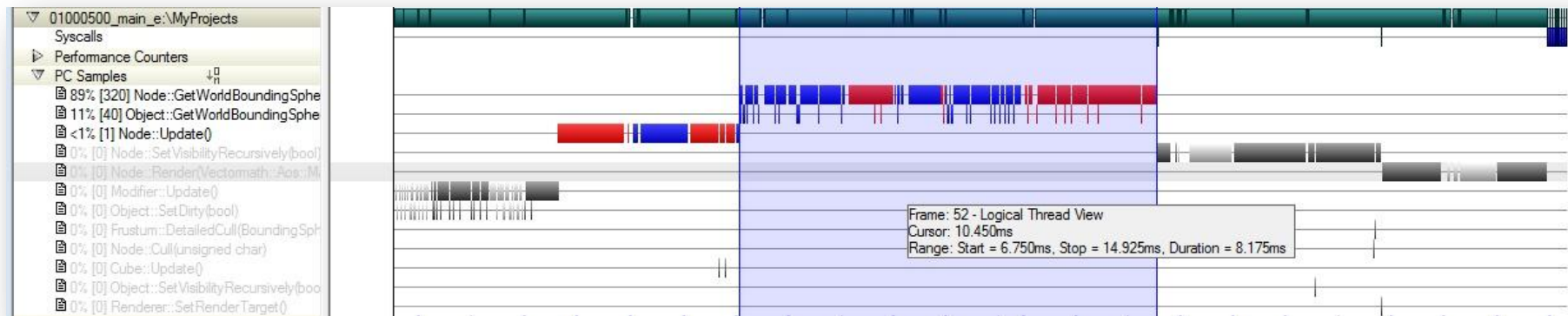- Hierarchical culling of tree
- Render method is empty

Why is it so slow?

~22ms

Look at GetWorldBoundingSphere()

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Samples can be a little misleading at the source code level

```
372    9    57  const BoundingSphere& Node::GetWorldBoundingSphere(const Matrix4& parentTransform)
  -    -    58
2998  49    59      if(!m_Dirty)
  -    -    60          return m_WorldBoundingSphere;
  -    -    61
  -    -    62      // if it was dirty, then we need to update the bounding volumes and transforms
2212  32    63      if(m_Parent)
  -    -    64          m_WorldTransform = parentTransform*m_Transform;
  -    -    65      else
  -    -    66          m_WorldTransform=m_Transform;
  -    -    67      // was dirty, so we need to recalculate the bounds of the children
 42    1    68      m_WorldBoundingSphere=BoundingSphere(); // zero it
  -    -    69
2179  46    70      for(std::vector<Object*>::const_iterator itr = m_Objects.begin();
  -    -    71          itr!= m_Objects.end();
  -    -    72          ++itr)
  -    -    73      {
9085 174   74          m_WorldBoundingSphere.ExpandBy((*itr)->GetWorldBoundingSphere(m_WorldTransform));
  -    -    75      }
123    3    76      m_Dirty=false;
 18    1    77      return m_WorldBoundingSphere;
314    6    78  }
  -    -    79
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Stalls due to the load 2 instructions earlier

```
372    9   57  const BoundingSphere&
       -   58  {
2998  49   59      if(!m_Dirty)
       -   60          return m_World
       -   61
       -   62      // if it was dirty
2212  32   63      if(m_Parent)
       -   64          m_WorldTransfc
       -   65      else
       -   66          m_WorldTransfc
       -   67      // was dirty, so
  42   1   68      m_WorldBoundingSp
       -   69
2179  46   70      for(std::vector<Ok
       -   71          itr!= m_Object
       -   72          ++itr)
       -   73      {
9085 174   74          m_WorldBoundir
       -   75      }
 123   3   76      m_Dirty=false;
  18   1   77      return m_WorldBour
 314   6   78  }
       -   79
```

```
  9    -      00012724 F8010100 std     r0,0x10
  8    -      00012728 DBE100E8 stfd    f31,0xE         
0001272C 386300B0 addi    r3,r3,0
  8    -      00012730 FB8100C0 std     r28,0xC0(r1)
 12    2      00012734 FBA100C8 std     r29,0x  8(r1)    PIPE
  9    1      00012738 881B00D4 lbz     r0,0xD4(r27)
 23    1      0001273C FBC100D0 std     r30,0xD0(r1)    PIPE
2989  48      00012740 2F800000 cmpwi   cr7,r0,0x0
       -      00012744 FBE100D8 std     r31,0xD8(r1)    PIPE
       -      00012748 419E026C beq     cr7,0x000129B4
       -      0001274C 801B00D8 lwz     r0,0xD8(r27)
 32    2      00012750 2F800000 cmpwi   cr7,r0,0x0     01 (0001274C) REG
 30    -      00012754 419E02A0 beq     cr7,0x000129F4  01 (00012750) REG
  1    -      00012758 397B0010 addi    r11,r27,0x10
  3    -      0001275C 7D2020CE lvx     v9,0,r4
  5    -      00012760 38E00010 li      r7,0x10
  1    -      00012764 79690020 clrldi  r9,r11,32      PIPE
       -      00012768 39000020 li      r8,0x20
 22    -      0001276C 38000030 li      r0,0x30        PIPE
  2    -      00012770 788A0020 clrldi  r10,r4,32
 19    -      00012774 7D8058CE lvx     v12,0,r11
  3    -      00012778 100004C4 vxor    v0,v0,v0
       -      0001277C 7D6938CE lvx     v11,r9,r7
  3    -      00012780 3B9B0050 addi    r28,r27,0x50
       -      00012784 7CA940CE lvx     v5,r9,r8
 18    -      00012788 7DA900CE lvx     v13,r9,r0
       -      0001278C 7B890020 clrldi  r9,r28,32      01 (00012780) REG
1344  19      00012790 10202A8C vspltw  v1,v5,0
       -      00012794 11012A8C vspltw  v8,v5,1        PIPE
  9    1      00012798 1040628C vspltw  v2,v12,0
```

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

```
 372    9   57  const BoundingSphere&
  -     -   58  {
2998   49   59      if(!m_Dirty)
  -     -   60          return m_Worl
  -     -   61
  -     -   62      // if it was dirty
2212   32   63      if(m_Parent)
  -     -   64          m_WorldTransfo
  -     -   65      else
  -     -   66          m_WorldTransfo
  -     -   67      // was dirty, so
  42     1   68      m_WorldBoundingSp
  -     -   69
2179   46   70      for(std::vector<Ob
  -     -   71          itr!= m_Object
  -     -   72          ++itr)
  -     -   73      {
9085  174   74          m_WorldBoundi
  -     -   75      }
 123    3   76      m_Dirty=false;
  18     1   77      return m_WorldBour
 314     6   78  }
  -     -   79
```

```
   9    -      00012724  F8010100  std    r0,0x100(r1)          PIPE
   8    -      00012728  DBE100E8  stfd   f31,0xE8(r1)
   -    -      0001272C  386300B0  addi   r3,r3,0xB0
   8    -      00012730  FB8100C0  std    r28,0xC0(r1)
  12    2      00012734  FBA100C8  std    r29,0xC8(r1)          PIPE
   9    1      00012738  881B00D4  lbz    r0,0xD4(r27)
  23    1      0001273C  FBC100D0  std    r30,0xD0(r1)          PIPE
2989   48      00012740  2F800000  cmpwi  cr7,r0,0x0
   -    -      00012744  FBE100D8  std    r31,0xD8
   -    -      00012748  419E026C  beq    cr7,0x0
   -    -      0001274C  801B00D8  lwz    r0,0xD8
  32    2      00012750  2F800000  cmpwi  cr7,r0,
  30    -      00012754  419E02A0  beq    cr7,0x0
   1    -      00012758  397B0010  addi   r11,r27
   3    -      0001275C  7D2020CE  lvx    v9,0,r4
   5    -      00012760  38E00010  li     r7,0x10
   1    -      00012764  79690020  clrldi r9,r11,32            PIPE
   -    -      00012768  39000020  li     r8,0x20
  22    -      0001276C  38000030  li     r0,0x30              PIPE
   2    -      00012770  788A0020  clrldi r10,r4,32
  19    -      00012774  7D8058CE  lvx    v12,0,r11
   3    -      00012778  100004C4  vxor   v0,v0,v0
   -    -      0001277C  7D6938CE  lvx    v11,r9,r7
   3    -      00012780  3B9B0050  addi   r28,r27,0x
   -    -      00012784  7CA940CE  lvx    v5,r9,r8
  18    -      00012788  7DA900CE  lvx    v13,r9,r0
   -    -      0001278C  7B890020  clrldi r9,r28,3            01 (00012780) REG
1344   19      00012790  10202A8C  vspltw v1,v5,0
   -    -      00012794  11012A8C  vspltw v6,v5,1              PIPE
   9    1      00012798  1040628C  vspltw v2,v12,0
```
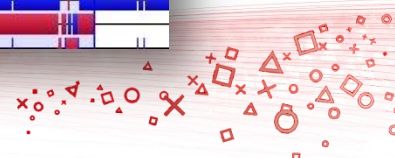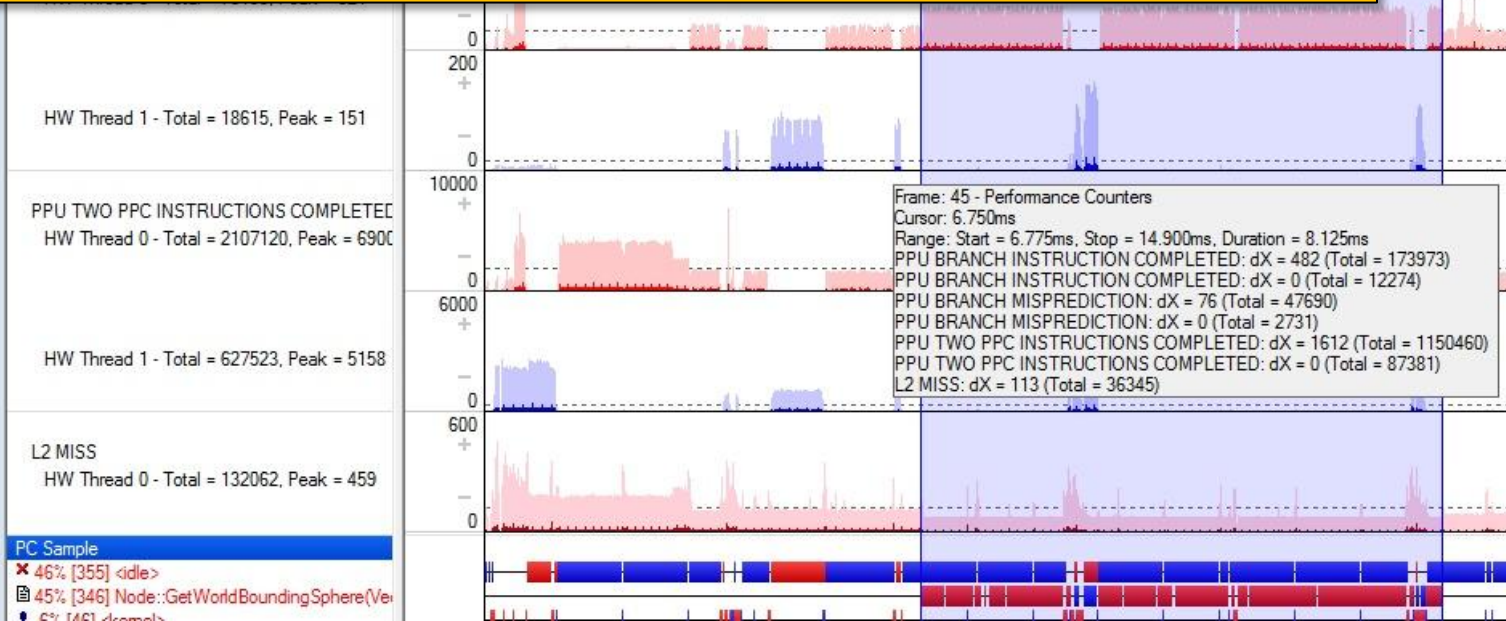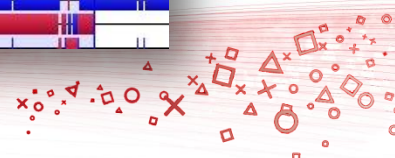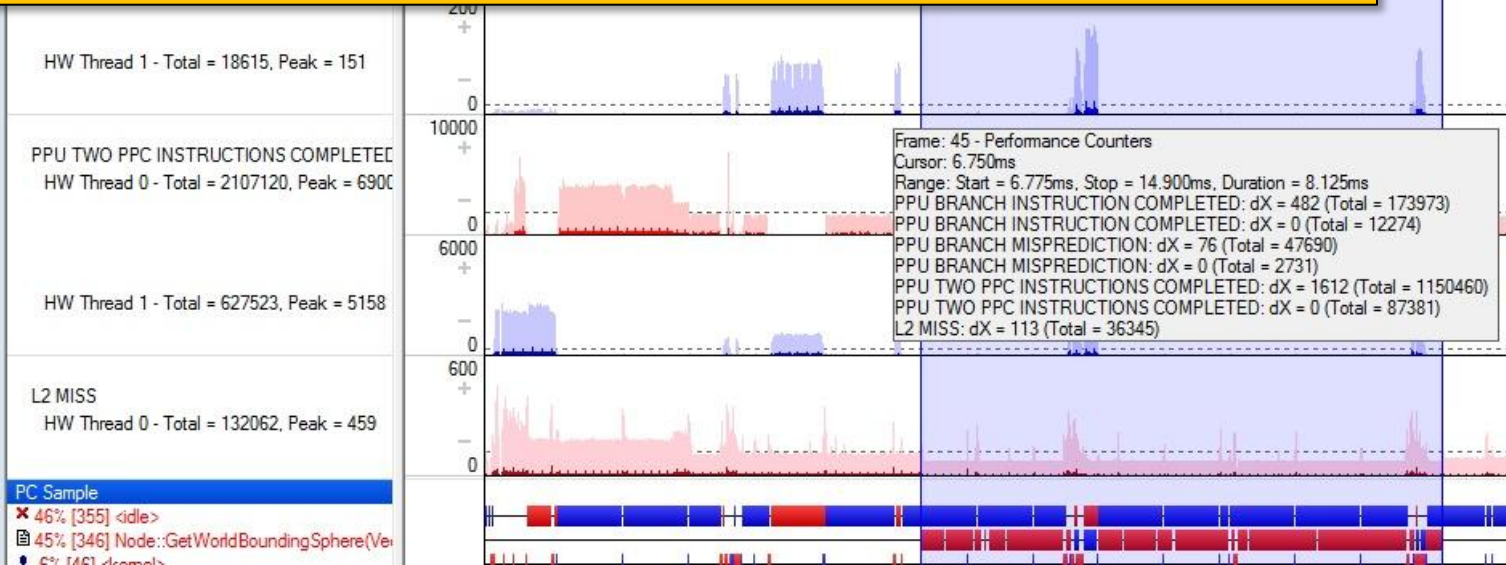
Similarly with the matrix multiply

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Some rough calculations

Branch Mispredictions: 50,421 @ 23 cycles each   ~= 0.36ms

HW Thread 1 - Total = 18615, Peak = 151

PPU TWO PPC INSTRUCTIONS COMPLETED
HW Thread 0 - Total = 2107120, Peak = 6900

HW Thread 1 - Total = 627523, Peak = 5158

L2 MISS
HW Thread 0 - Total = 132062, Peak = 459

PC Sample
✗ 46% [355] <idle>
🗎 45% [346] Node::GetWorldBoundingSphere(Ve
● 6% [46] <kernel>

Frame: 45 - Performance Counters
Cursor: 6.750ms
Range = Start = 6.775ms, Stop = 14.900ms, Duration = 8.125ms
PPU BRANCH INSTRUCTION COMPLETED: dX = 482 (Total = 173973)
PPU BRANCH INSTRUCTION COMPLETED: dX = 0 (Total = 12274)
PPU BRANCH MISPREDICTION: dX = 76 (Total = 47690)
PPU BRANCH MISPREDICTION: dX = 0 (Total = 2731)
PPU TWO PPC INSTRUCTIONS COMPLETED: dX = 1612 (Total = 1150460)
PPU TWO PPC INSTRUCTIONS COMPLETED: dX = 0 (Total = 87381)
L2 MISS: dX = 113 (Total = 36345)

Some rough calculations

Branch Mispredictions: 50,421 @ 23 cycles each   ~= 0.36ms
L2 Cache misses: 36,345 @ 400 cycles each   ~= 4.54ms

HW Thread 1 - Total = 18615, Peak = 151

PPU TWO PPC INSTRUCTIONS COMPLETED
HW Thread 0 - Total = 2107120, Peak = 6900

HW Thread 1 - Total = 627523, Peak = 5158

L2 MISS
HW Thread 0 - Total = 132062, Peak = 459

Frame: 45 - Performance Counters
Cursor: 6.750ms
Range: Start = 6.775ms, Stop = 14.900ms, Duration = 8.125ms
PPU BRANCH INSTRUCTION COMPLETED: dX = 482 (Total = 173973)
PPU BRANCH INSTRUCTION COMPLETED: dX = 0 (Total = 12274)
PPU BRANCH MISPREDICTION: dX = 76 (Total = 47690)
PPU BRANCH MISPREDICTION: dX = 0 (Total = 2731)
PPU TWO PPC INSTRUCTIONS COMPLETED: dX = 1612 (Total = 1150460)
PPU TWO PPC INSTRUCTIONS COMPLETED: dX = 0 (Total = 87381)
L2 MISS: dX = 113 (Total = 36345)

PC Sample
✗ 46% [355] <idle>
🖹 45% [346] Node::GetWorldBoundingSphere(Ve

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

- From Tuner, ~ 3 L2 cache misses per object
  - These cache misses are mostly sequential (more than 1 fetch from main memory can happen at once)
  - Code/cache miss/code/cache miss/code…

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Slow memory is the problem here

- How can we fix it?

- And still keep the same functionality and interface?

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

The first step

- Use homogenous, sequential sets of data

```
class Object
{
    // <methods removed for clarity>
    Matrix4 *m_Transform;
    Matrix4 *m_WorldTransform;
    BoundingSphere *m_BoundingSphere;
    BoundingSphere *m_WorldBoundingSphere;
    char* m_Name;
    bool m_Dirty;
    Object* m_Parent;
};
```

vtbl	Matrix4*	Matrix4*	Bsphere*
Bsphere*	char*	bool	Object*
vector			

```
class Node : public Object
{
    // methods removed for clarity
    std::vector<Object*> m_Objects;
    bool m_IsVisible;
};
```

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

Homogeneous Sequential Data

vtbl	Matrix4*	Matrix4*	Bsphere*
Bsphere*	char*	bool	Object*
vector			

World transforms					Local Transforms					World Bounding Spheres					Local Bounding Spheres			
Matrix4					Matrix4					Vector3					Vector3			
										Vector3					Vector3			
										Vector3					Vector3			
										Vector3					Vector3			
Matrix4					Matrix4					Vector3					Vector3			
										Vector3					Vector3			
										Vector3					Vector3			
										Vector3					Vector3			
Matrix4					Matrix4												
Matrix4					Matrix4												

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Generating Contiguous Data

- Use custom allocators
    - Minimal impact on existing code
- Allocate contiguous
    - Nodes
    - Matrices
    - Bounding spheres

Performance

Object Oriented   Contiguous

Node Update: 6.01 / 4.03
Transform & Update Bounding Spheres: 7.07 / 5.75
Culling Pass: 3.74 / 1.79
Node Render: 2.79 / 1.33
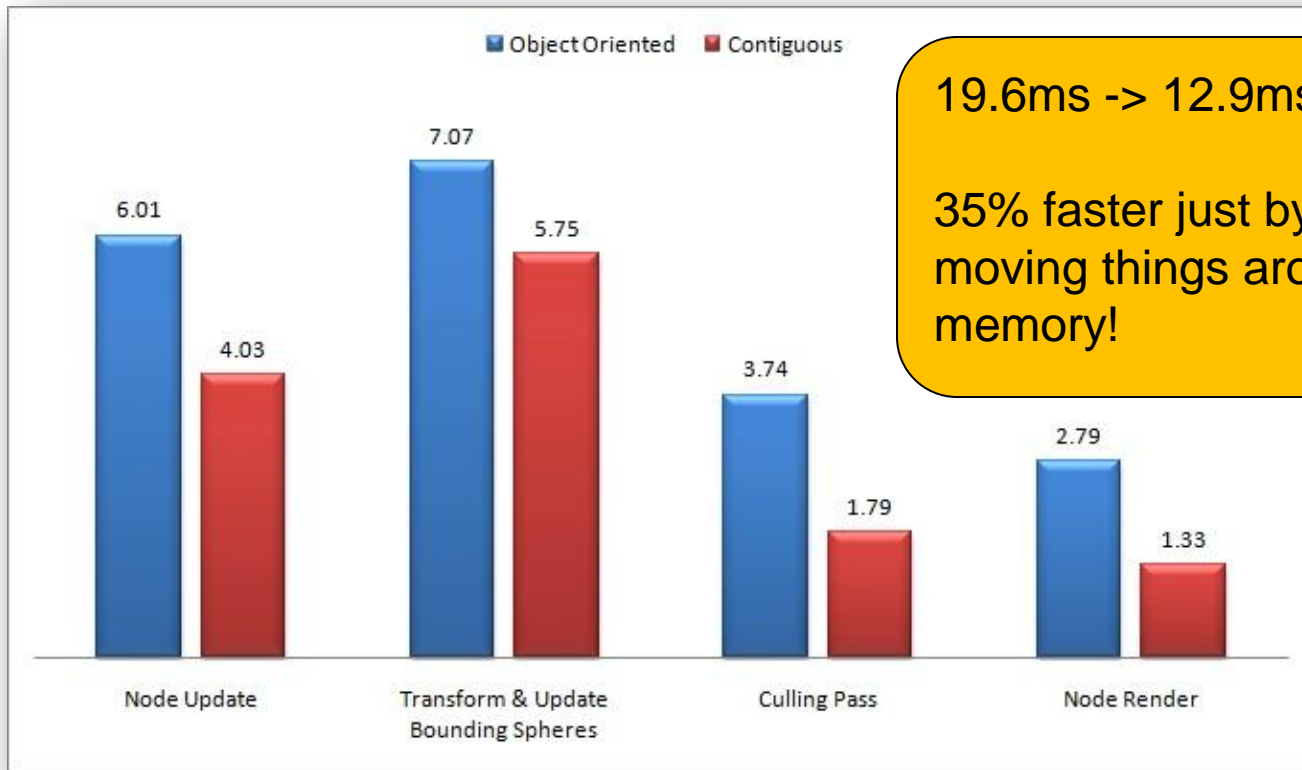
19.6ms -> 12.9ms

35% faster just by moving things around in memory!

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

What next?

- Process data in order
- Use implicit structure for hierarchy
  - Minimise to and fro from nodes.
- Group logic to optimally use what is already in cache.
- Remove regularly called virtuals.

Hierarchy

We start with a parent Node

Node

Hierarchy

Node

Node

Which has children nodes
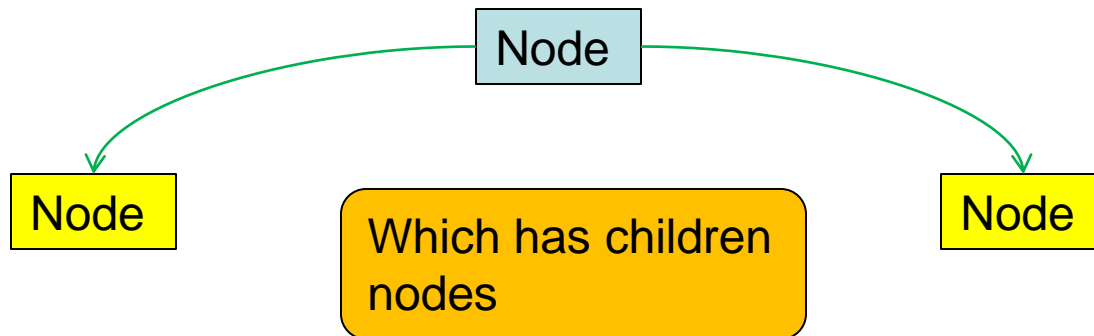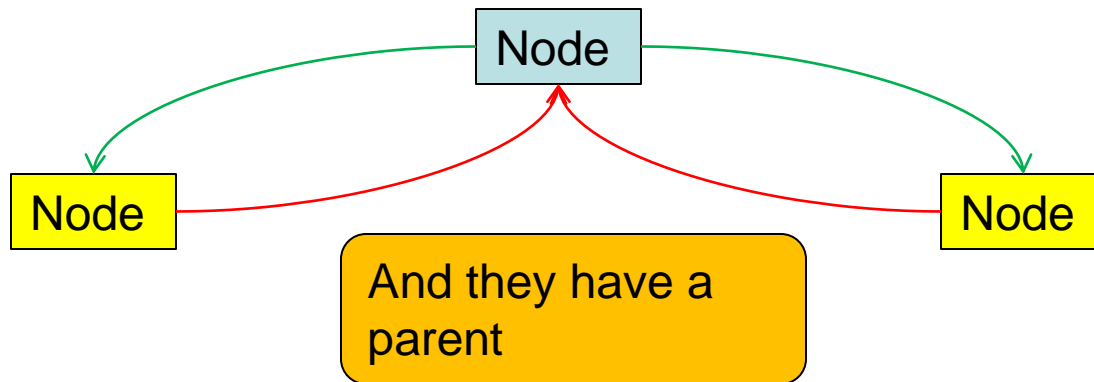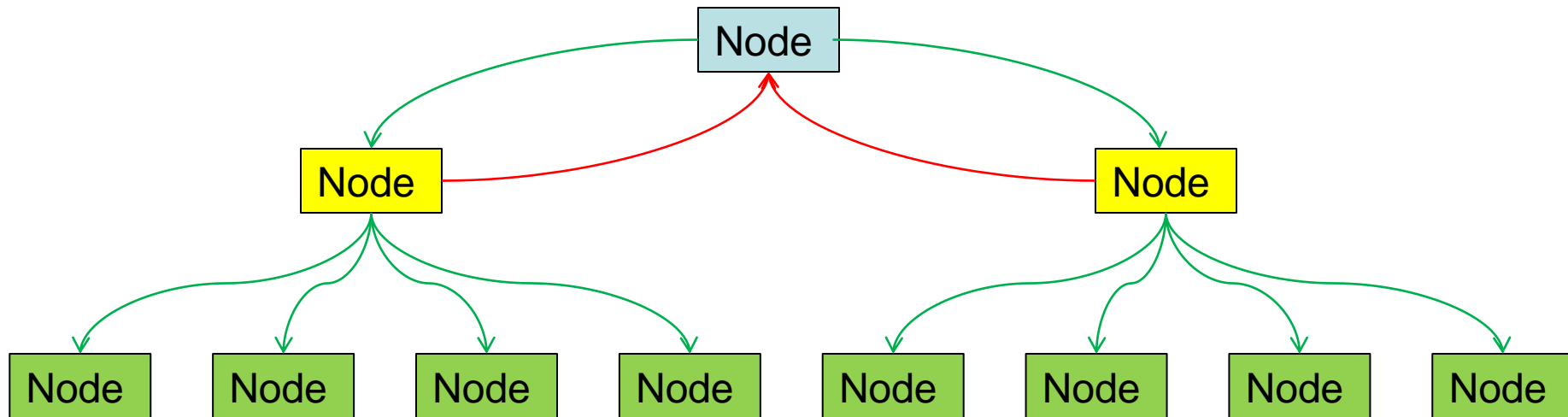
Node

Hierarchy

And they have children
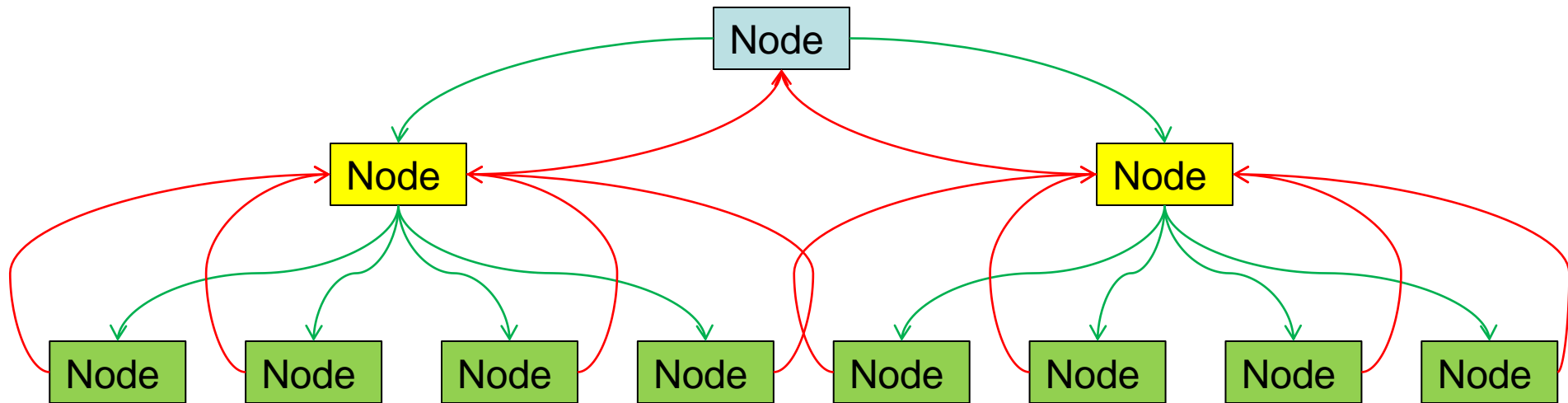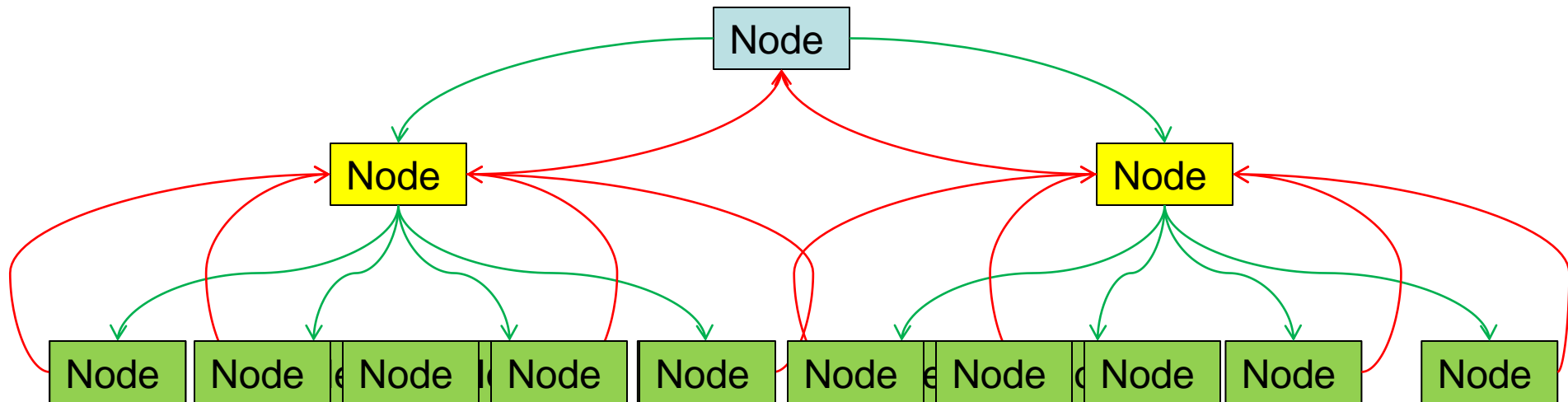
Hierarchy

A lot of this information can be inferred

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Hierarchy

Node

Node Node

Node Node Node Node Node Node Node Node

Use a set of arrays, one per hierarchy level

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

Hierarchy

Node

:2

Node   Node

:4

:4

Node Node Node Node Node Node Node Node

Parent has 2 children

Children have 4 children

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
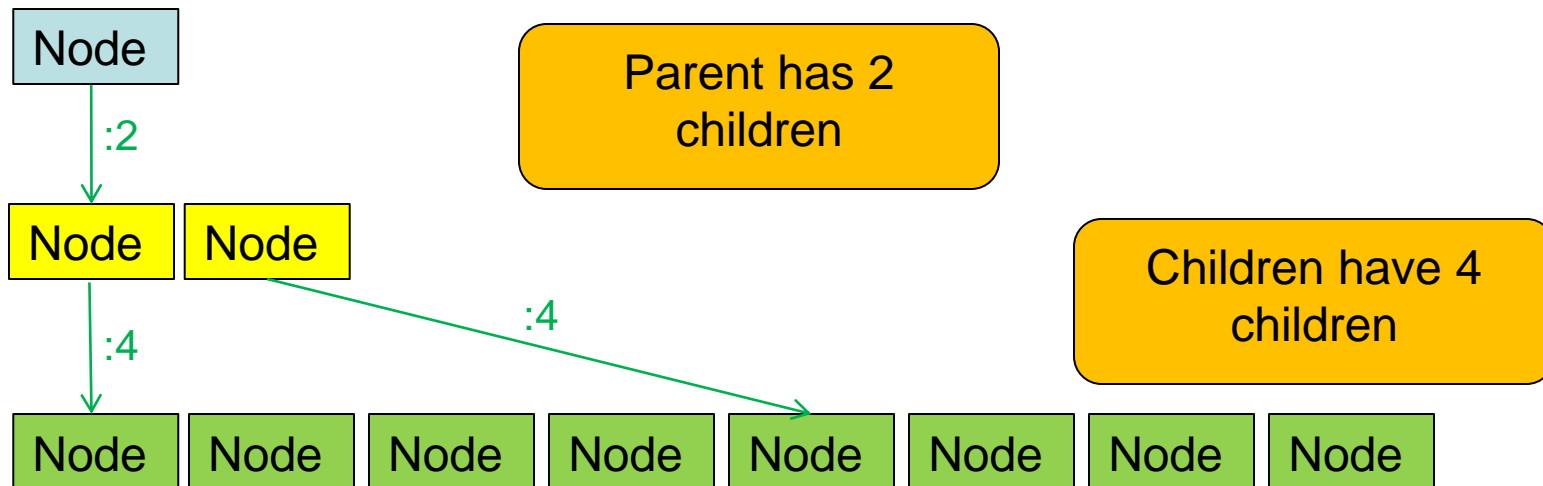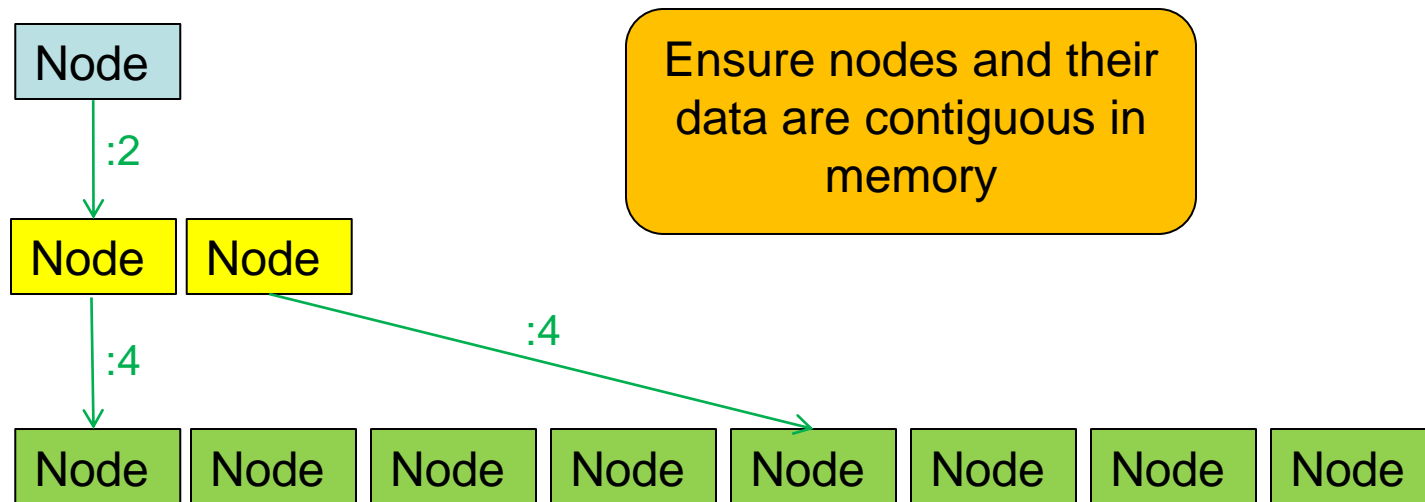
- Make the processing global rather than local
  - Pull the updates out of the objects.
    - No more virtuals
  - Easier to understand too – all code in one place.

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Need to change some things…

- ## OO version
  - – Update transform top down and expand WBS bottom up

Add bounding sphere of child

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

- Hierarchical bounding spheres pass info up

- Transforms cascade down

- Data use and code is 'striped'.

  – Processing is alternating

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

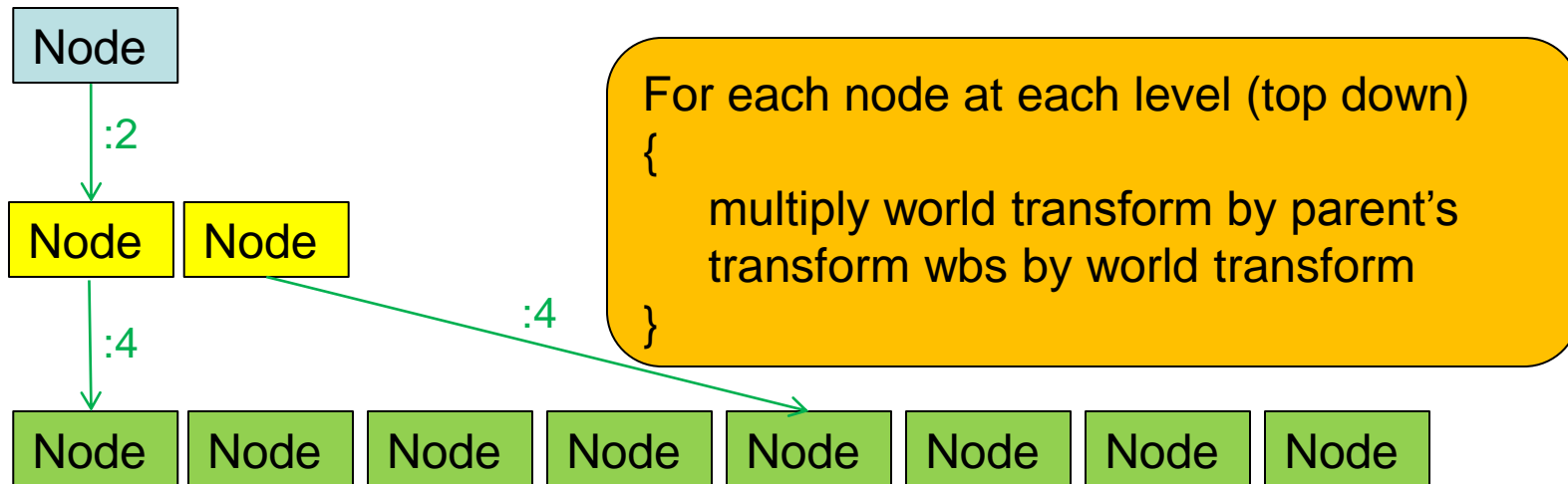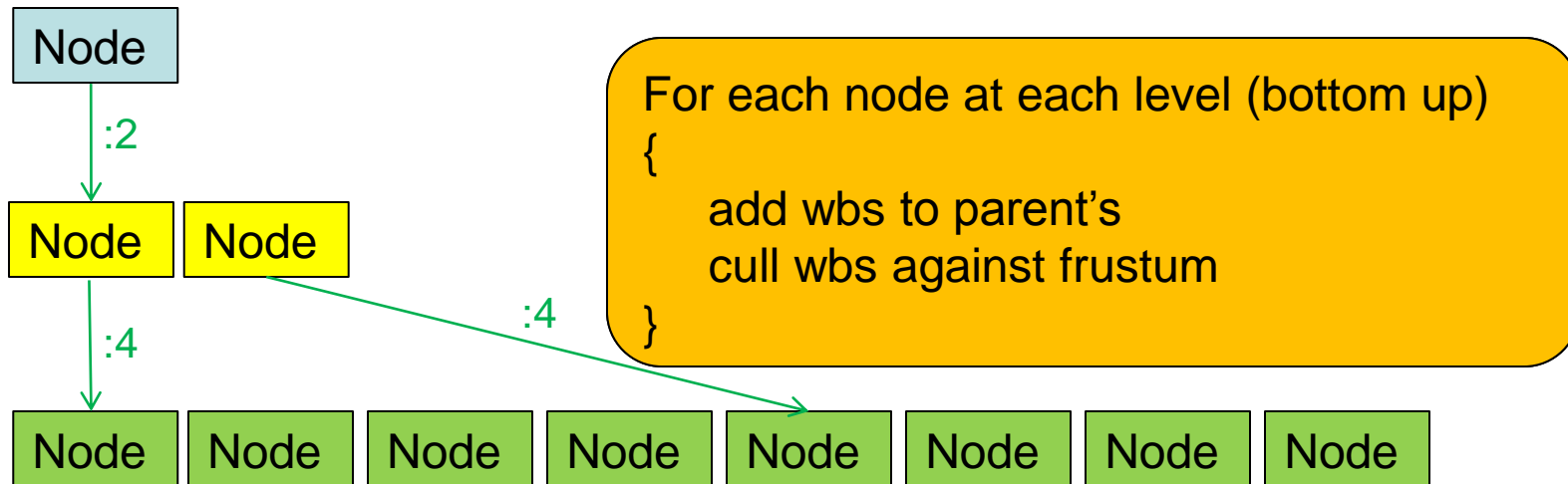# Conversion to linear

- To do this with a 'flat' hierarchy, break it into 2 passes
    - Update the transforms and bounding spheres(from top down)
    - Expand bounding spheres (bottom up)

Transform and BS updates

For each node at each level (top down)
{

    multiply world transform by parent's
    transform wbs by world transform

}

Update Transform and Bounding Sphere

How many children nodes to process

```
for(int j=0;j<size;j++)
{
    const int innerSize = parent->m_Objects.size();
    const Matrix4 *parentTransform = parent->m_WorldTransform;

    j+=innerSize;
    for(int k=0;k<innerSize;k++, wmat++, mat++, bs++, wbs++)
    {
        *wmat = (*parentTransform)*(*mat);
        *wbs = bs->Transform(wmat);
    }
    parent++;
}
```

Update Transform and Bounding Sphere

```
for(int j=0;j<size;j++)
{
    const int innerSize = parent->m_Objects.size();
    const Matrix4 *parentTransform = parent->m_WorldTransform;

    j+=innerSize;
    for(int k=0;k<innerSize;k++, wmat++, mat++, bs++, wbs++)
    {
        *wmat = (*parentTransform)*(*mat);
        *wbs = bs->Transform(wmat);
    }
    parent++;
}
```

For each child, update transform and bounding sphere

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

Update Transform and Bounding Sphere


```
for(int j=0;j<size;j++)
{
    const int innerSize = parent->m_Objects.size
    const Matrix4 *parentTransform = parent->m_WorldT

    j+=innerSize;
    for(int k=0;k<innerSize;k++, wmat++, mat++, bs++, wbs++)
    {
        *wmat = (*parentTransform)*(*mat);
        *wbs = bs->Transform(wmat);
    }
    parent++;
}
```

Note the contiguous arrays

So, what's happening in the cache?

Load parent and its transform

Parent
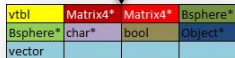
Unified L2 Cache

```
const int innerSize = parent->m_Objects.size();
const Matrix4 *parentTransform = parent->m_WorldTransform;
```

Parent Data

Childrens' Data

Load child transform and set world transform

Parent

Parent Data

```
for(int k=0;k<innersize;k++, wmat++, mat++, bs++, wbs++)
{
    *wmat = (*parentTransform)*(*mat);
    *wbs = bs->Transform(wmat);
}
```

Childrens' Data

Unified L2 Cache

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY

Load child BS and set WBS

Parent

Unified L2 Cache

```
for(int k=0;k<innersize;k++, wmat++, mat++, bs++, wbs++)
{
    *wmat = (*parentTransform)*(*mat);
    *wbs = bs->Transform(wmat);
}
```

Parent Data

Childrens' Data

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

Load child BS and set WBS

Parent

Next child is calculated with
no extra cache misses !

Unified L2 Cache

```
for(int k=0;k<inner...
{
    *wmat = (*parentTransform)*(*mat);
    *wbs = bs->Transform(wmat);
}
```

Parent Data

Childrens' Data

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

Load child BS and set WBS

Parent

The next 2 children incur 2 cache misses in total

Unified L2 Cache

```
for(int k=0;k<inner
{
    *wmat = (*parentTransform)*(*mat);
    *wbs = bs->Transform(wmat);
}
```

Parent Data

Childrens' Data

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY COMPUTER ENTERTAINMENT

Prefetching

Parent

vtbl	Matrix4*	Matrix4*	Bsphere*
Bsphere*	char*	bool	Object*
vector			

```
for(int k=0;k<inner
{
    *wmat = (*paren
    *wbs = bs->Tran
}
```
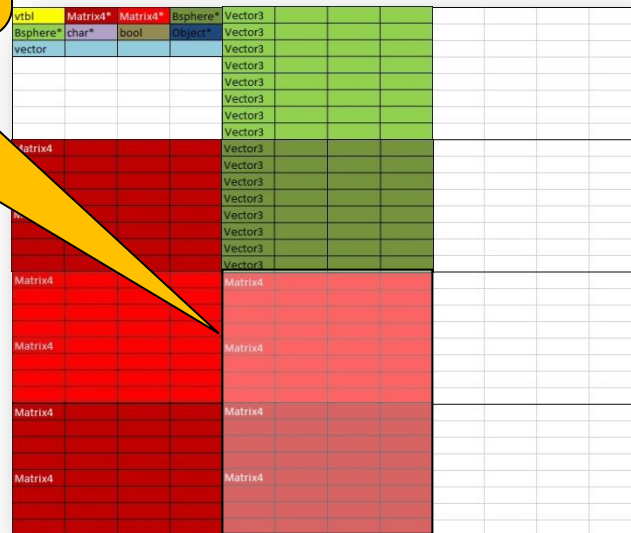
Parent Data

Because all data is linear, we can predict what memory will be needed in ~400 cycles and prefetch

Childrens' Data

Unified L2 Cache

vtbl	Matrix4*	Matrix4*	Bsphere*	Vector3			
Bsphere*	char*	bool	Object*	Vector3			
vector				Vector3			

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe
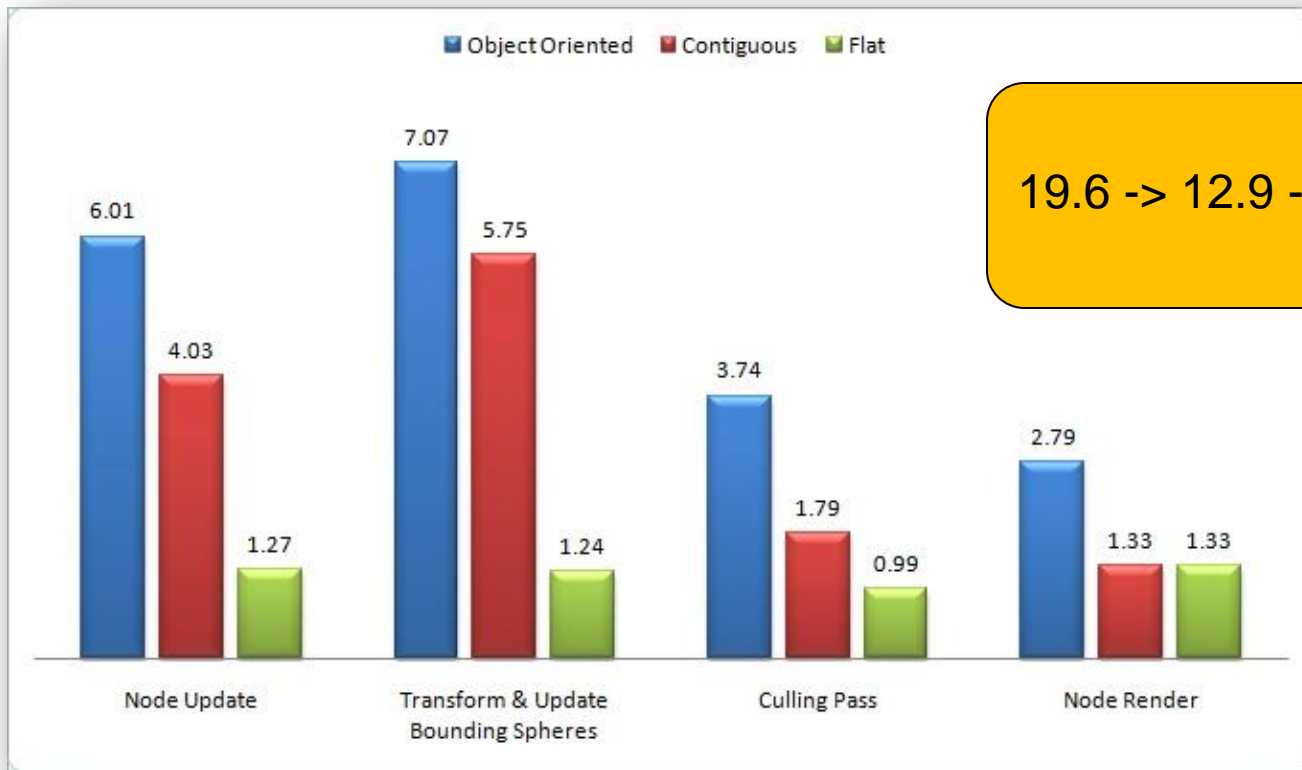
SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT
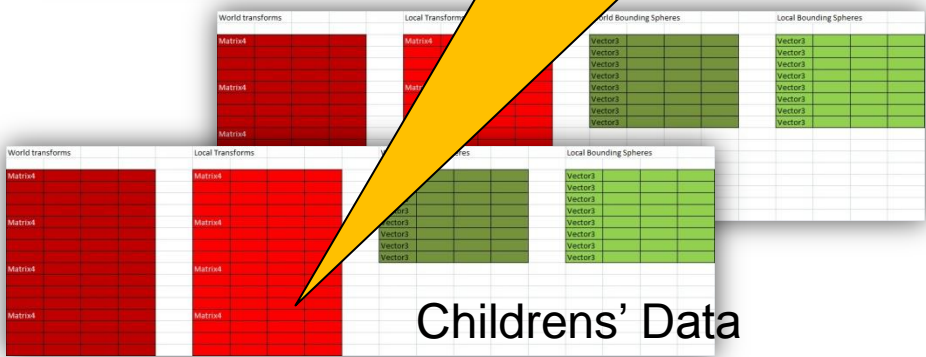
- Tuner scans show about 1.7 cache misses per node.

- But, these misses are much more frequent
  - Code/cache miss/cache miss/code
  - Less stalling

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Performance

19.6 -> 12.9 -> 4.8ms

PS3
PlayStation3

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

Prefetching

- Data accesses are now predictable
- Can use prefetch (dcbt) to warm the cache
  - Data streams can be tricky
  - Many reasons for stream termination
  - Easier to just use dcbt blindly
    - (look ahead x number of iterations)

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
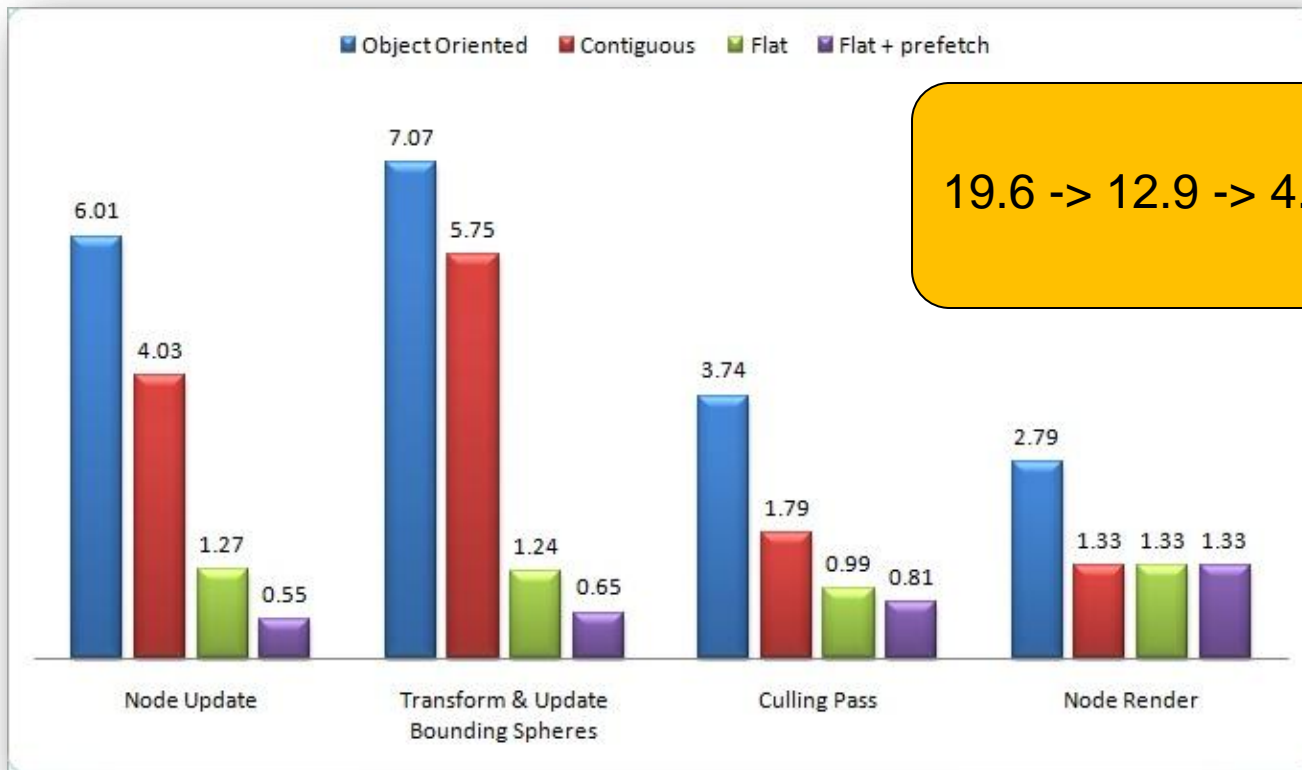Development
SONY
COMPUTER
ENTERTAINMENT

Prefetching example

- Prefetch a predetermined number of iterations ahead

- Ignore incorrect prefetches

```
for(int i=0;i<m_CurrentNumTransforms;i++)
{
    __dcbt(m_Transforms[i+20]);
    *m_Transforms[i] = (*m_Transforms[i])*(*m_Transform);
}
```

Performance

19.6 -> 12.9 -> 4.8 -> 3.3ms

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
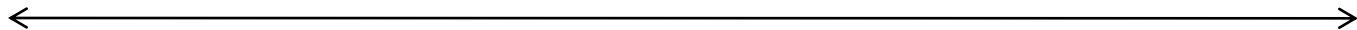Development

SONY
COMPUTER
ENTERTAINMENT

A Warning on Prefetching

- This example makes very heavy use of the cache

- This can affect other threads' use of the cache

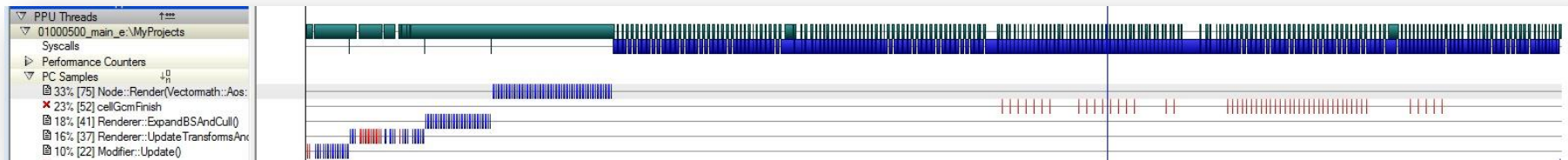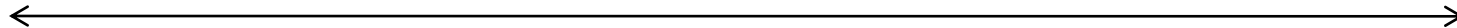 – Multiple threads with heavy cache use may thrash the cache
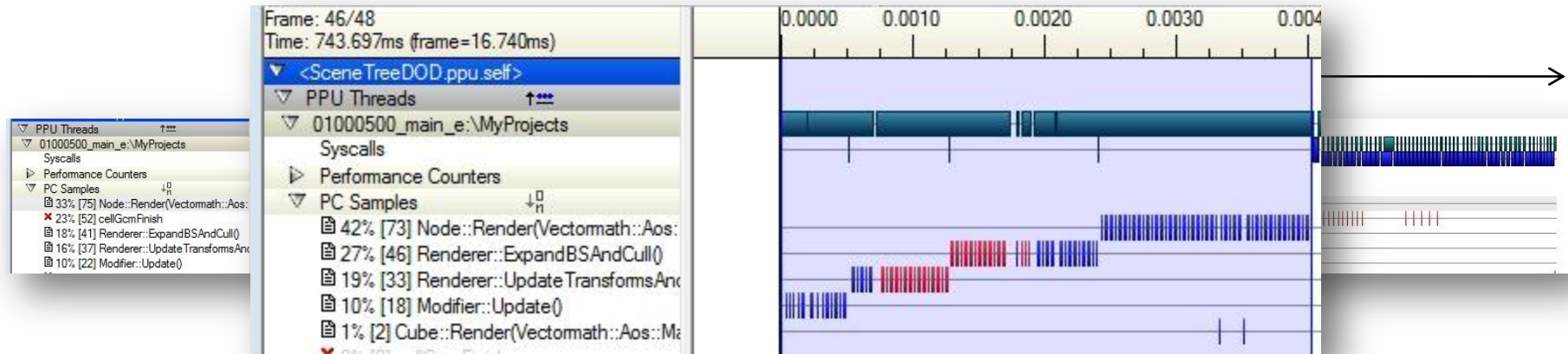
The old scan

~22ms

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

The new scan

~16.6ms

Up close

Looking at the code (samples)
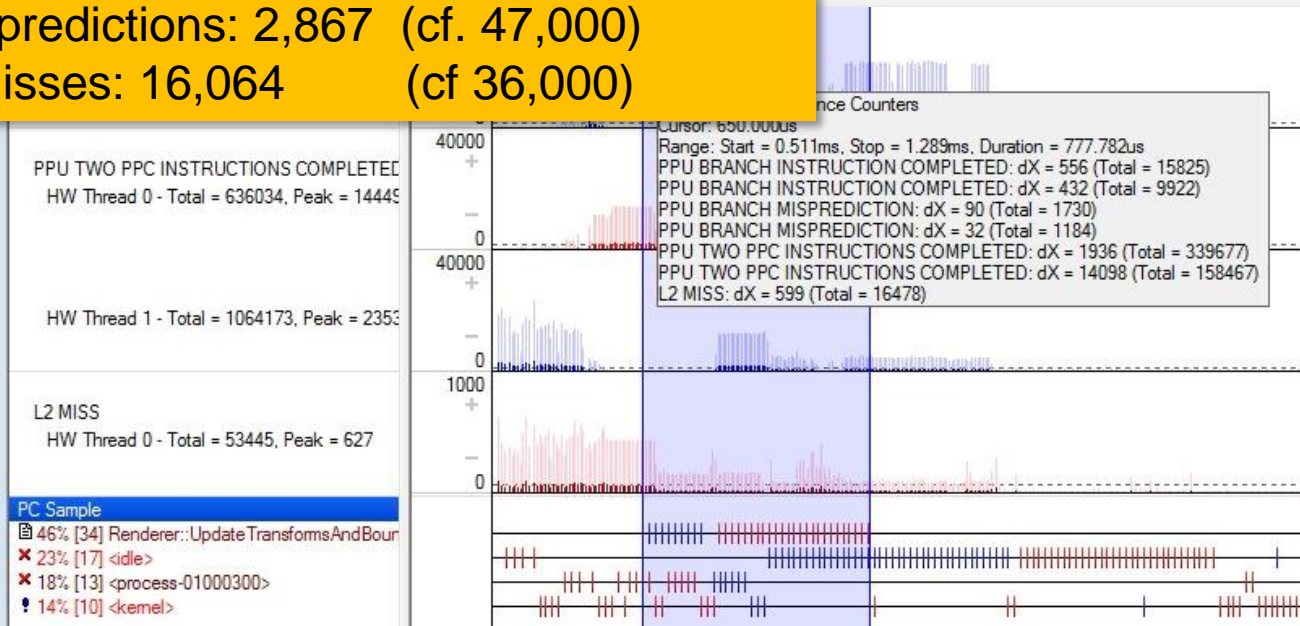
```
   -   - 449
 151   3 450          const Node* parent = (Node*)node->m_Parent;
   -   - 451          // iterate through all the matrices at this level, multiplying them by their parent
   8   - 452          for(int j=0;j<size;j++)//,node++, wmat++, mat++, wbs++, bs++)
   -   - 453          {
   -   - 454              const int innerSize = parent->m_Objects.size();
  25   - 455              const Matrix4 *parentTransform = parent->m_WorldTransform;
   -   - 456
   2   - 457              j+=innerSize;
 372   9 458              for(int k=0;k<innerSize;k++, wmat++, mat++, bs++, wbs++, node++)
   -   - 459              {
 233   2 460                  __dcbt(wmat+lookAhead);
 146   3 461                  __dcbt(mat+lookAhead);
 317   5 462                  __dcbt(bs+lookAhead);
 117   4 463                  __dcbt(wbs+lookAhead);
   -   - 464                  *wmat = (*parentTransform)*(*mat);
   -   - 465 //               *wmat = (*node->m_Parent->m_WorldTransform)*(*mat);
   -   - 466
  32   1 467                  *wbs = bs->Transform(wmat);
   -   - 468              }
   -   - 469              parent++;
```

Performance counters

Branch mispredictions: 2,867   (cf. 47,000)
L2 cache misses: 16,064   (cf 36,000)

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY

In Summary

- Just reorganising data locations was a win

- Data + code reorganisation= dramatic improvement.

- + prefetching equals even more WIN.

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

OO is not necessarily EVIL

- Be careful not to design yourself into a corner
- Consider data in your design
  - Can you decouple data from objects?
  - …code from objects?
- Be aware of what the compiler and HW are doing

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development
SONY
COMPUTER
ENTERTAINMENT

Its all about the memory

- Optimise for data first, then code.
  - Memory access is probably going to be your biggest bottleneck
- Simplify systems
  - KISS
  - Easier to optimise, easier to parallelise

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Homogeneity

- Keep code and data homogenous
    - Avoid introducing variations
    - Don't test for exceptions – sort by them.
- Not everything needs to be an object
    - If you must have a pattern, then consider using Managers

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Remember

- You are writing a GAME
  - You have control over the input data
  - Don't be afraid to preformat it – drastically if need be.

- Design for specifics, not generics (generally).

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

Data Oriented Design Delivers

- Better performance

- Better realisation of code optimisations

- Often simpler code

- More parallelisable code

LEVEL 4 PUBLIC
Sony Computer Entertainment Europe

SCEE
Research &
Development

SONY
COMPUTER
ENTERTAINMENT

The END

- Questions?