Heterogeneous Parallelism at Microsoft

Herb Sutter

1975-2005

Put a **computer** on every desk, in every home, in every pocket.

The free lunch is so over

2005-2011

Put a **parallel supercomputer** on every desk, in every home, in every pocket.

Welcome to the jungle

2011-201x

Put a **heterogeneous supercomputer** on every desk, in every home, in every pocket.

mainstream, *df.*:

commercially available to millions

commercially affordable for millions

commercially programmable by millions

note: everything in "the mainstream" starts out in "the exotic"

GUIs ✓ *objects* ✓ *parallelism* ✓

mainstream trends

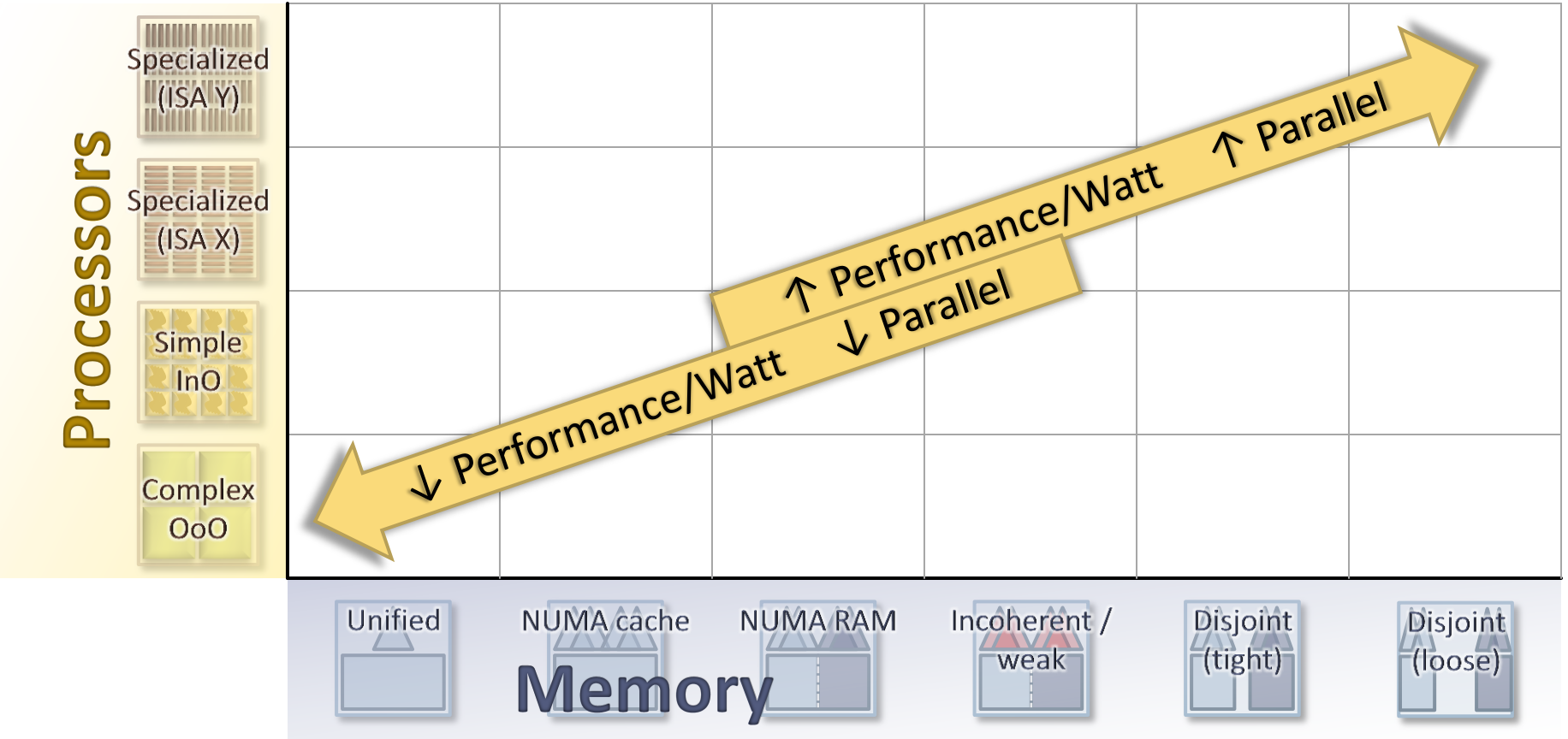
multicore CPUs

GPGPU

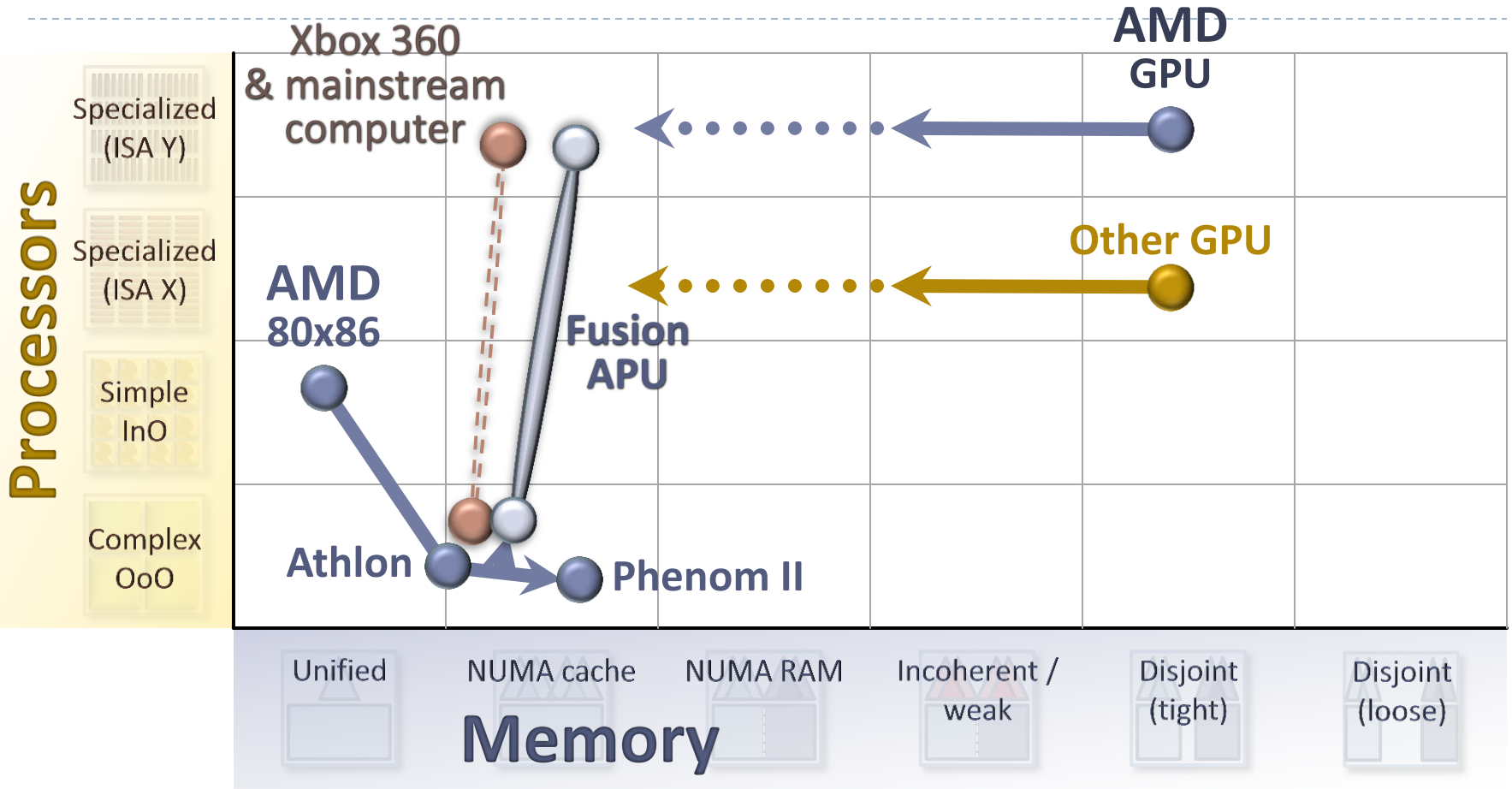
cloud IaaS/HaaS

Heterogeneous parallel computing

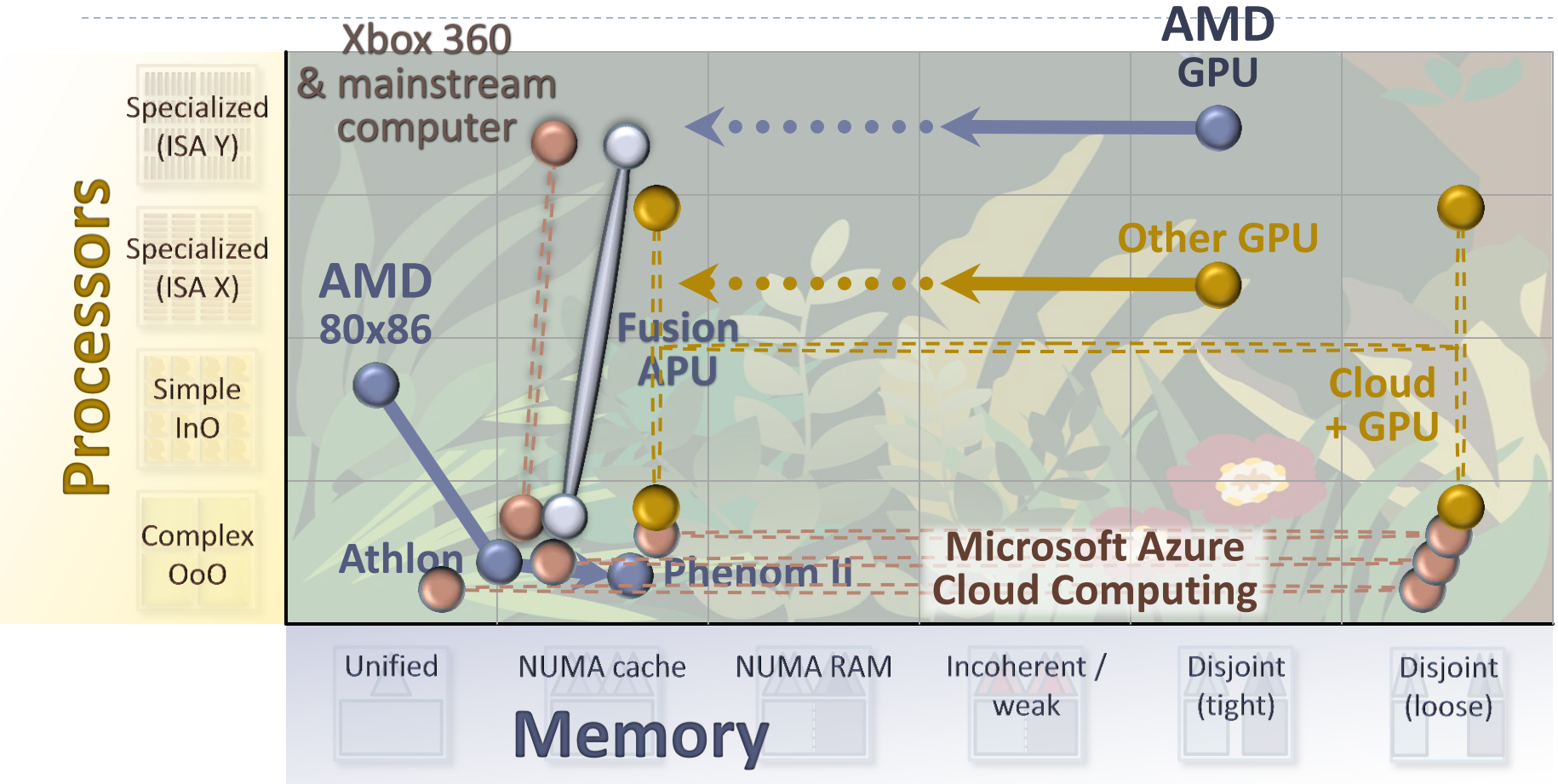
Charting the Landscape



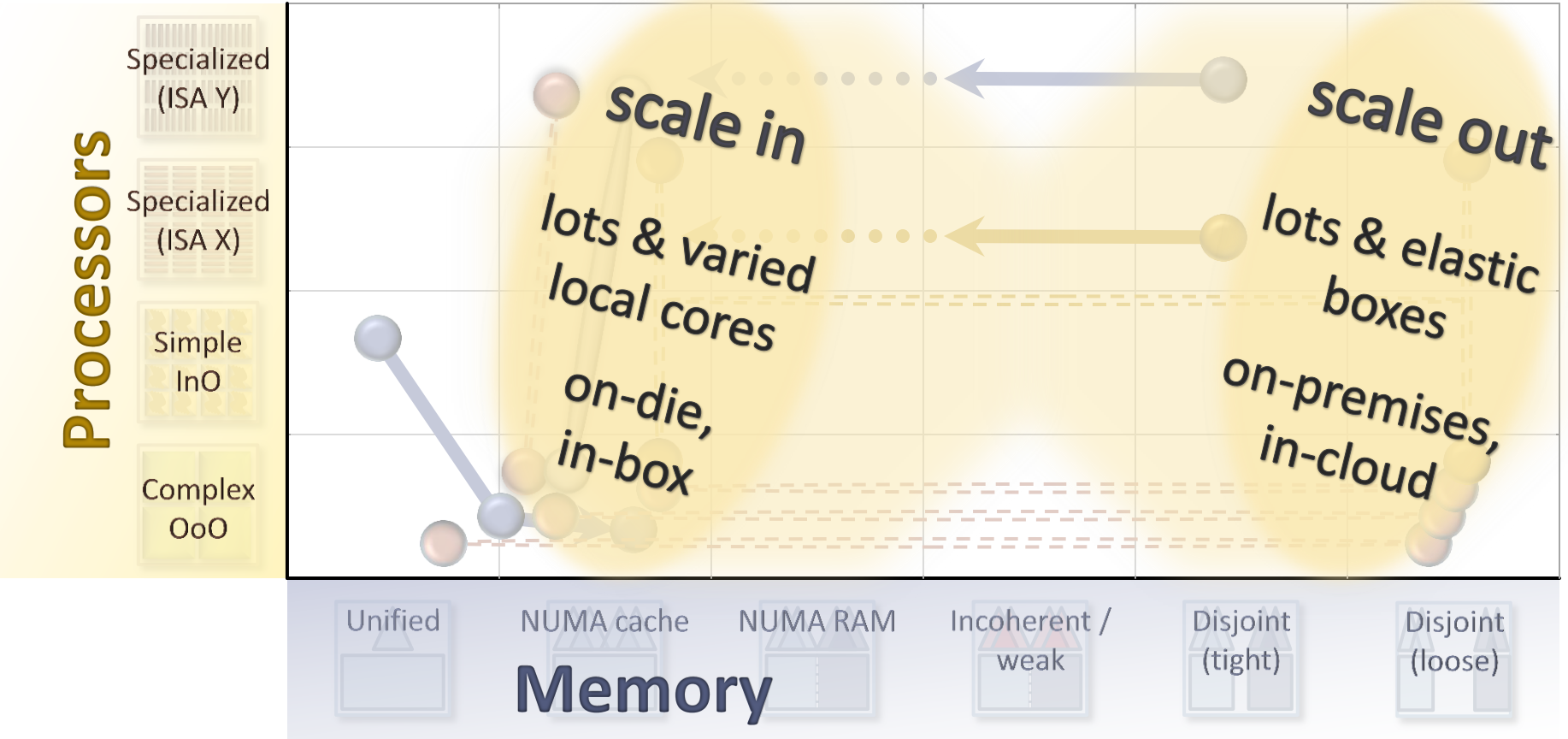
Hardware


Xbox 360
& mainstream
computer
AMD
GPU
Other GPU
AMD
80x86
Fusion
APU
Athlon
Phenom II
Processors
Specialized
(ISA Y)
Specialized
(ISA X)
Simple
InO
Complex
OoO
Unified
NUMA cache
NUMA RAM
Incoherent /
weak
Disjoint
(tight)
Disjoint
(loose)
Memory

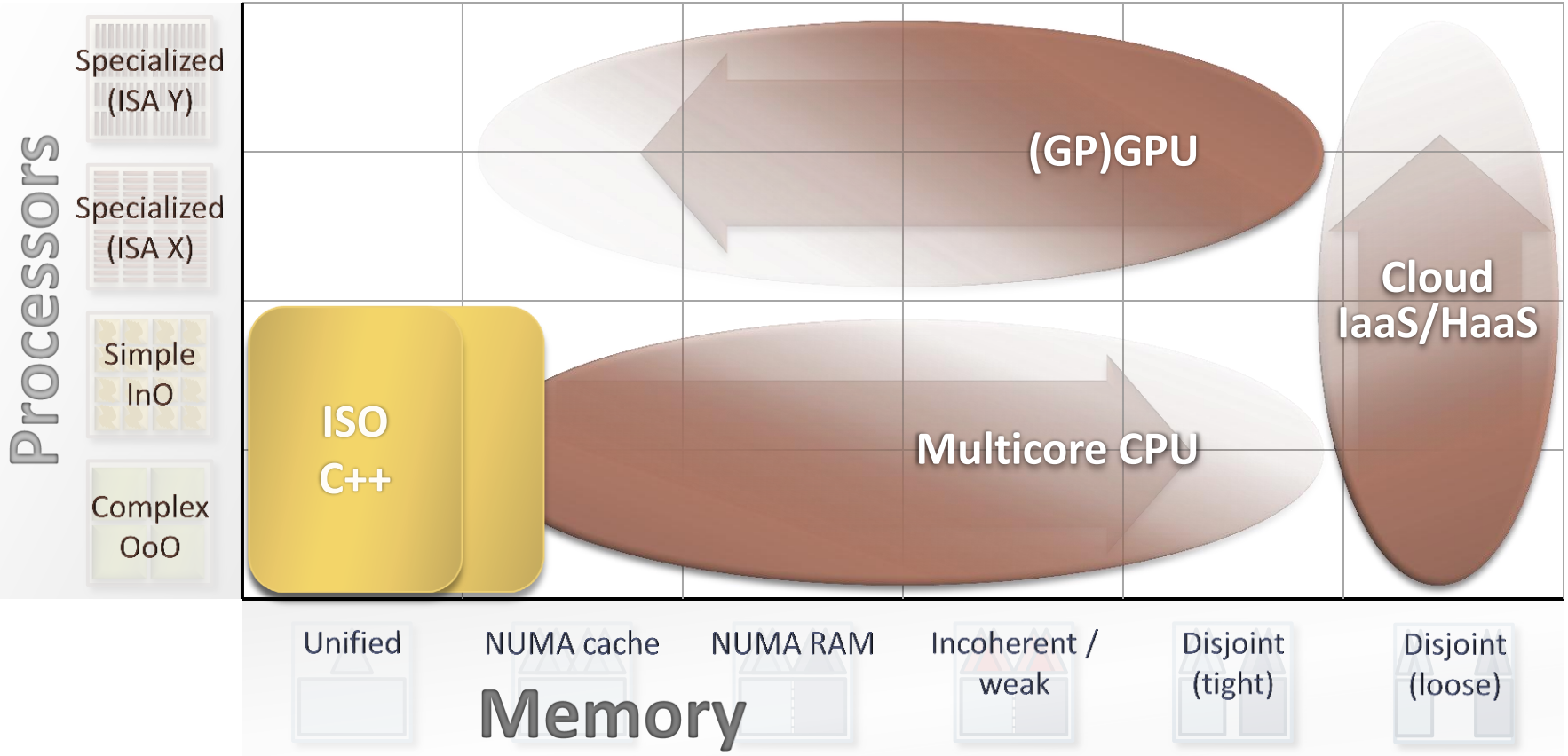
Hardware


Processors
Specialized (ISA Y)
Specialized (ISA X)
Simple InO
Complex OoO
Xbox 360 & mainstream computer
AMD GPU
Other GPU
AMD 80x86
Fusion APU
Cloud + GPU
Athlon
Phenom II
Microsoft Azure Cloud Computing
Unified
NUMA cache
NUMA RAM
Incoherent / weak
Disjoint (tight)
Disjoint (loose)
Memory

Hardware Evolution



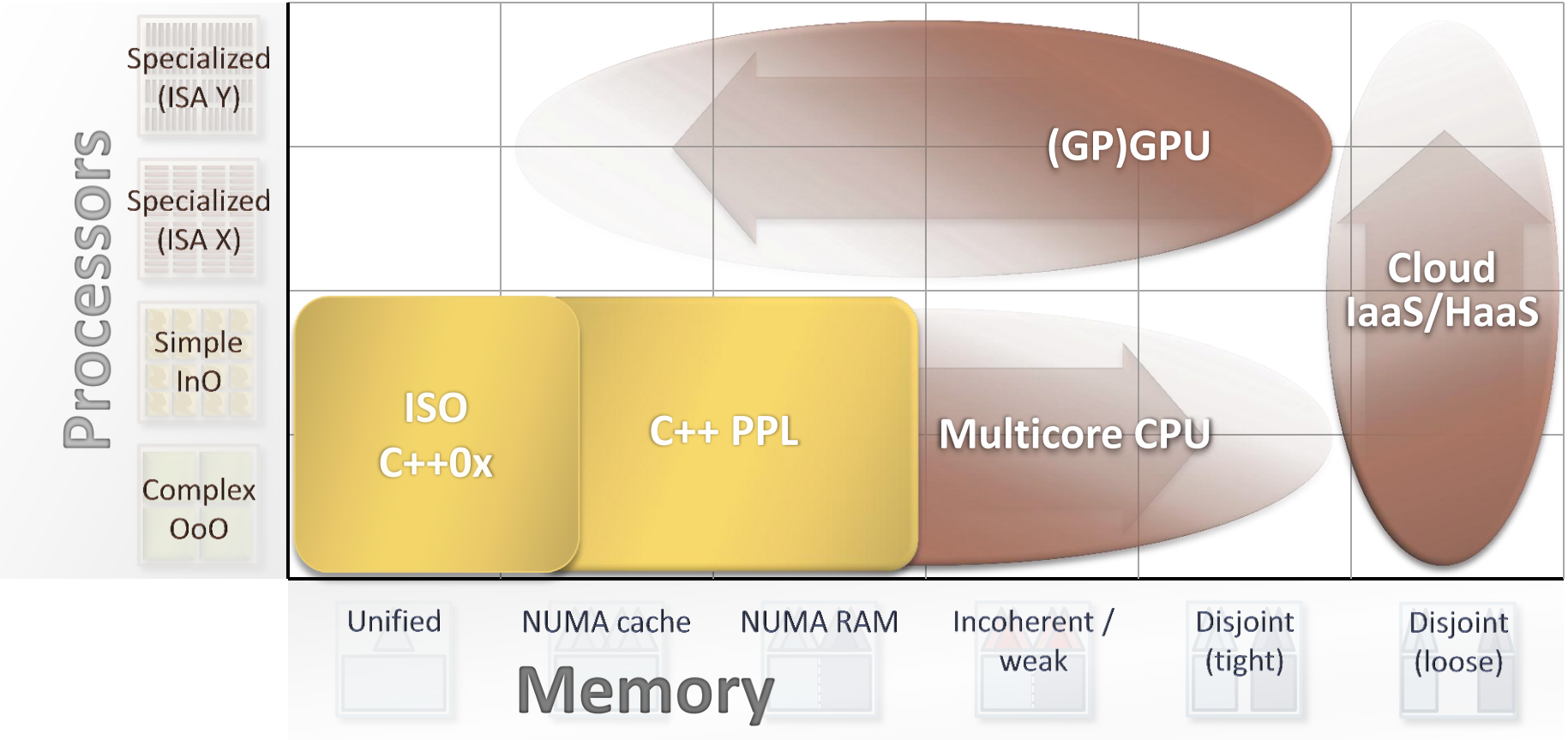
Programming Models & Languages



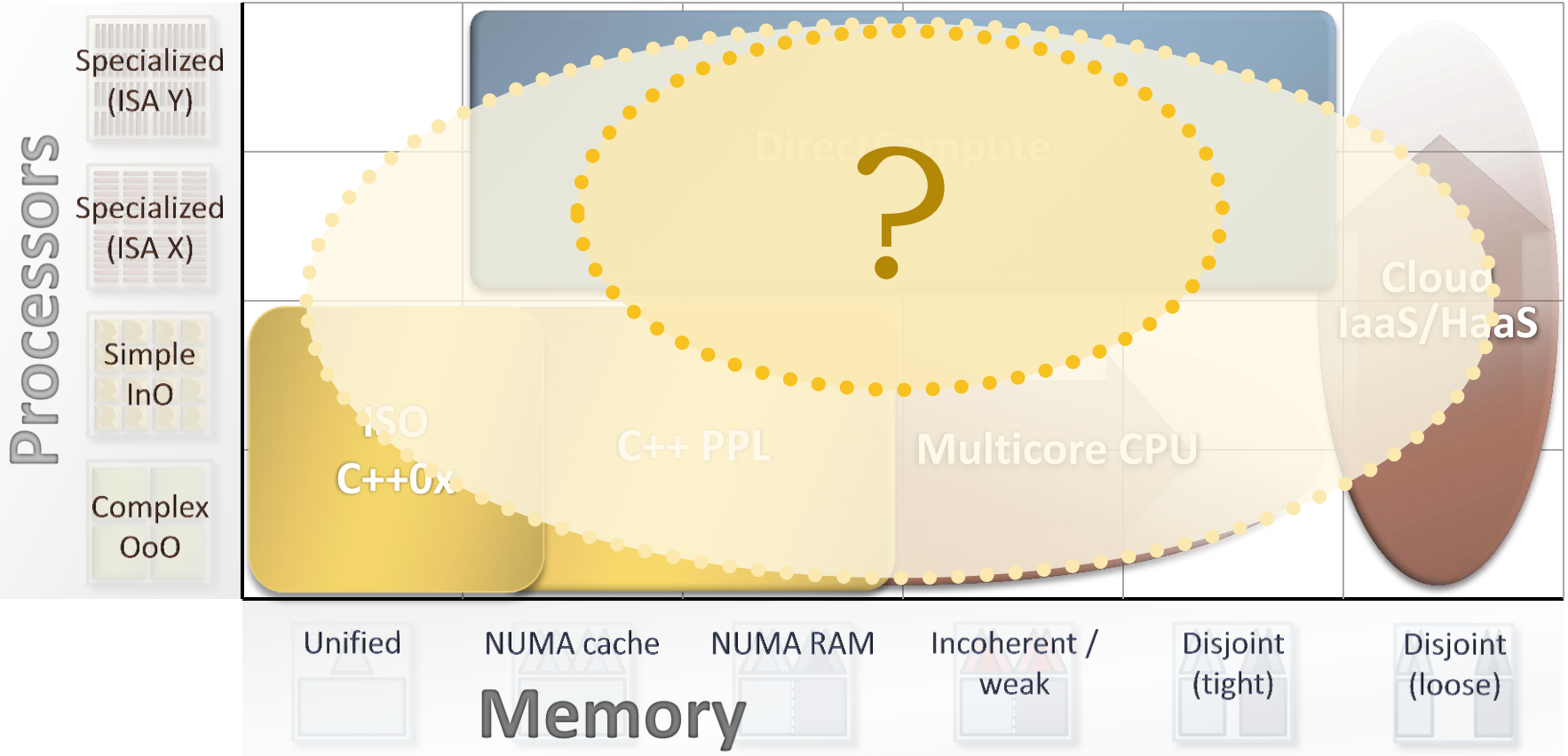
News Flash


Slashdot
stories
recent
C++ the Clear Winner In Google's Language Performance Tests
Google has released a research paper that suggests C++ is the best-performing programming language in the market.
The internet giant implemented a compact algorithm in four languages – C++, Java, Scala and its own programming language Go – and then benchmarked results to find "factors of difference".
"We find that in terms of performance, C++ wins out by a large margin," the paper says.
Further reading
› Surge in C++ demand

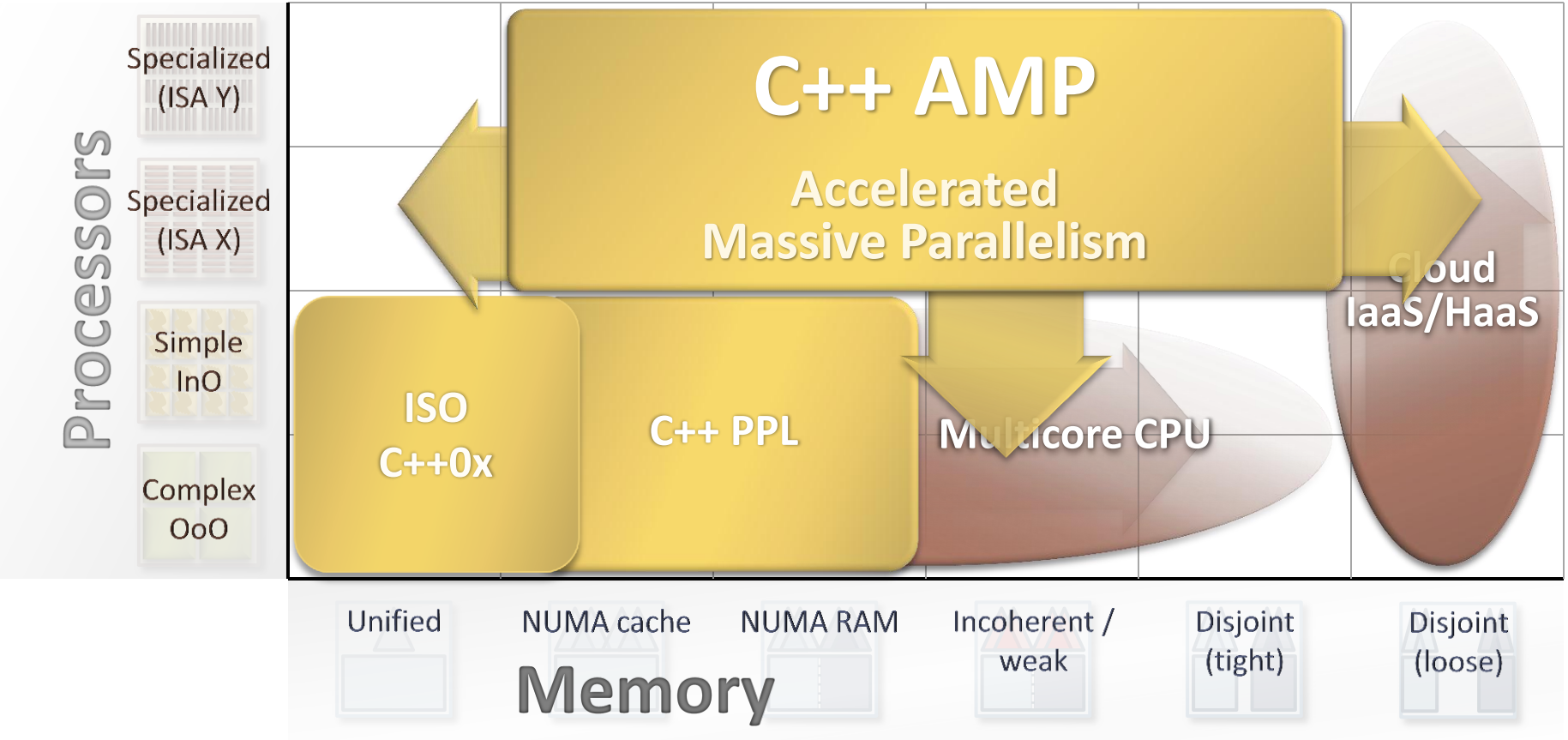
Programming Models & Languages



Programming Models & Languages



Programming Models & Languages



Matrix Multiply

Convert this (serial loop nest)

```
void MatrixMult( float* C, const vector<float>& A,  const vector<float>& B,
                 int M, int N, int W )
{
  for (int y = 0; y < M; y++)
    for (int x = 0; x < N; x++) {
      float sum = 0;
      for(int i = 0; i < W; i++)
        sum += A[y*W + i] * B[i*N + x];
      C[y*N + x] = sum;
    }
}
```

Matrix Multiply

Convert this (serial loop nest)

```
void Mat

{
  for (int
    for (ir
      floa
      for(
        su
      C[y*
    }
}
```

… to this (parallel loop, CPU or GPU)

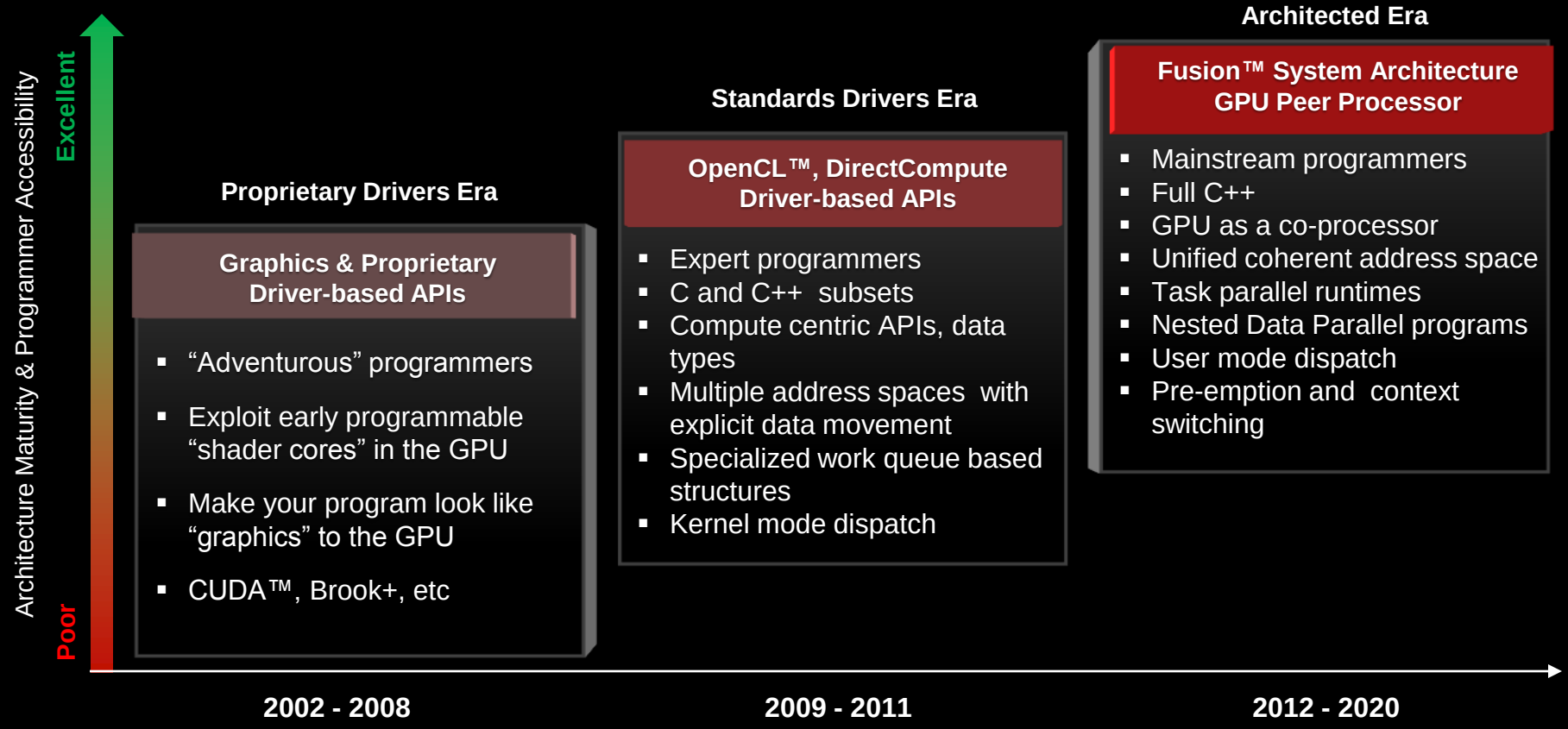
```
void MatrixMult( float* C, const vector<float>& A,  const vector<float>& B,
                 int M, int N, int W )
{
  array_view<const float,2> a(M,W,A), b(W,N,B);
  array_view<writeonly<float>,2> c(M,N,C);

  parallel_for_each( c.grid, [=](index<2> idx) restrict(direct3d) {
    float sum = 0;
    for(int i = 0; i < a.x; i++)
      sum += a(idx.y, i) * b(i, idx.x);
    c[idx] = sum;
  } );
}
```

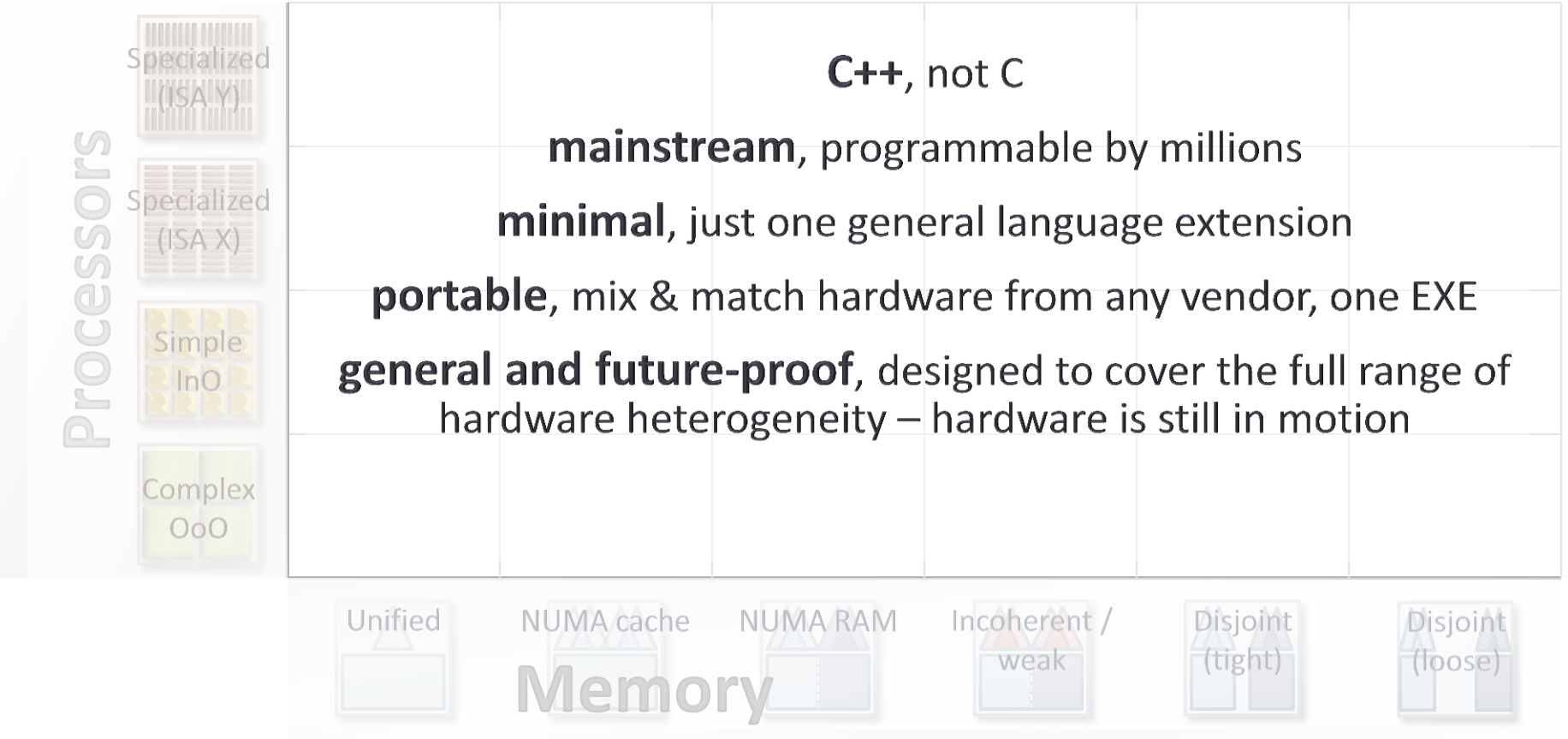
Demo

Daniel Moth
Program Manager
Parallel Computing Platform

EVOLUTION OF HETEROGENEOUS COMPUTING

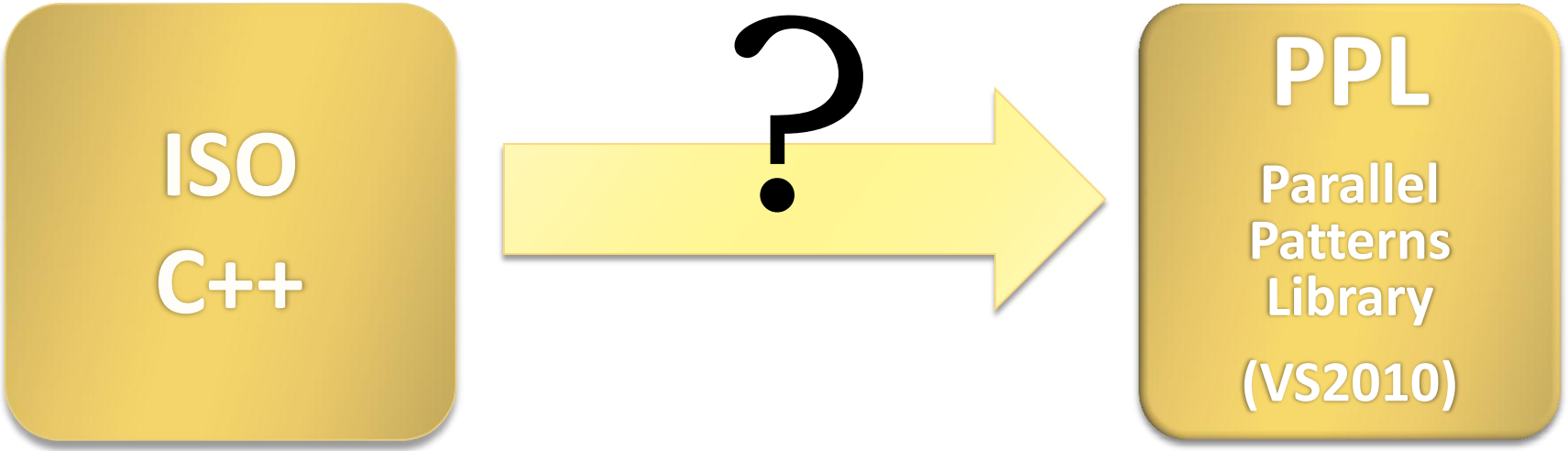


Why C++ AMP?


Processors
Specialized (ISA Y)
Specialized (ISA X)
Simple InO
Complex OoO
C++, not C
mainstream, programmable by millions
minimal, just one general language extension
portable, mix & match hardware from any vendor, one EXE
general and future-proof, designed to cover the full range of hardware heterogeneity – hardware is still in motion
Unified
NUMA cache
NUMA RAM
Incoherent / weak
Disjoint (tight)
Disjoint (loose)
Memory

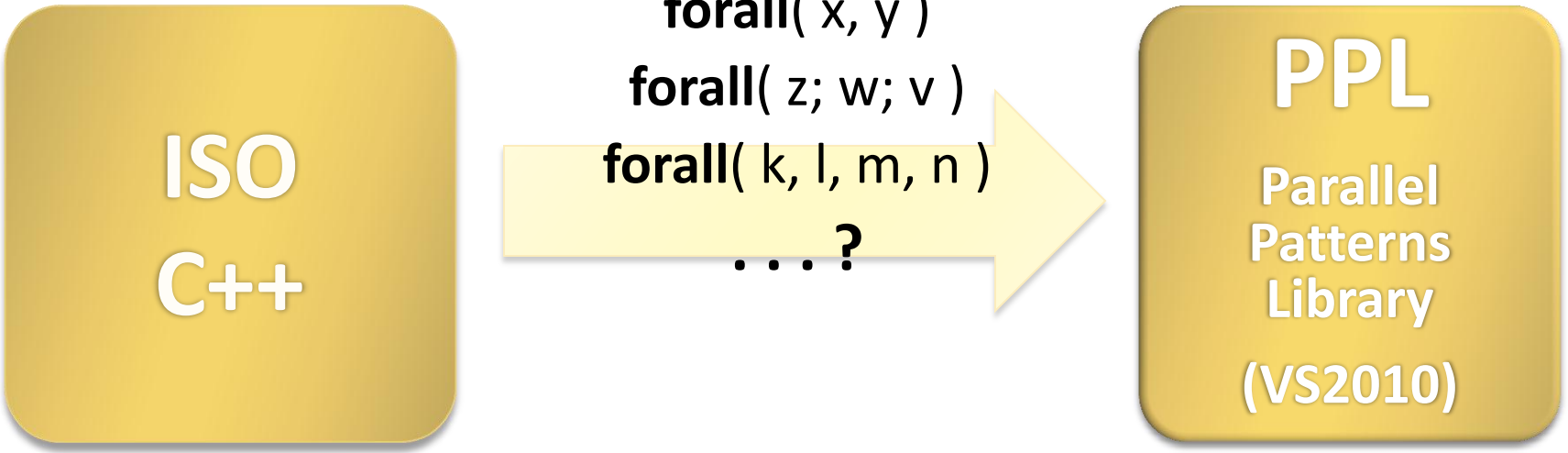
Language Design: Parallelism Phase 1

Single-core to multi-core



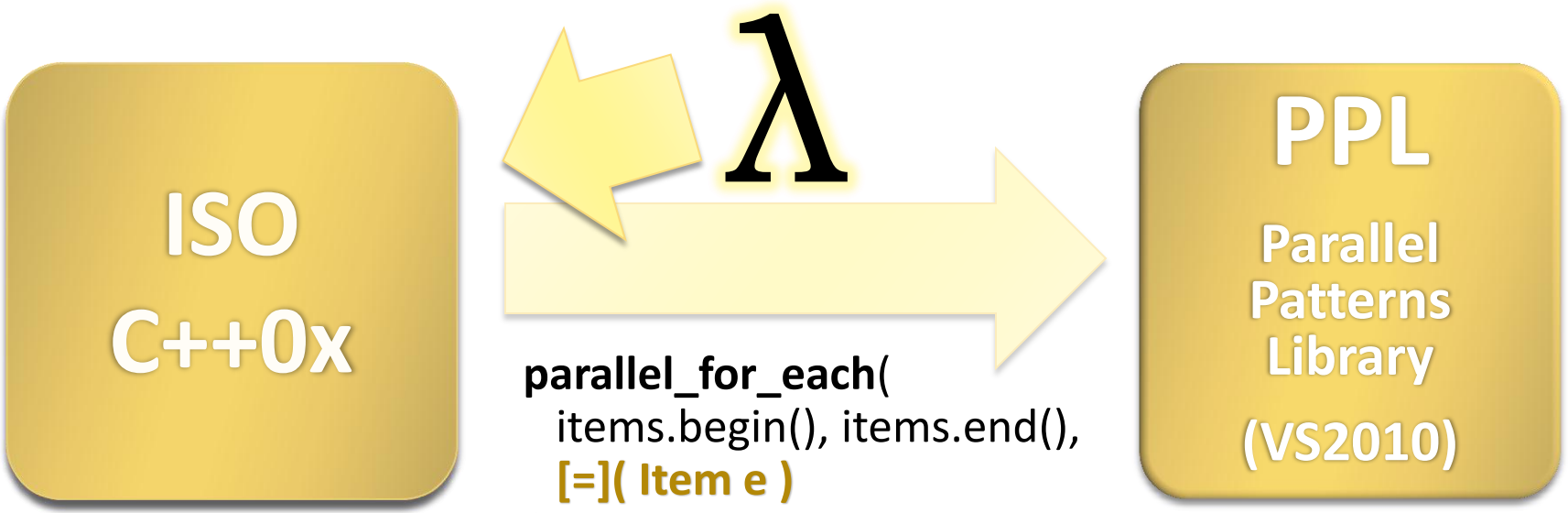
Language Design: Parallelism Phase 1

Single-core to multi-core



Language Design: Parallelism Phase 1

Single-core to multi-core




```
parallel_for_each(
  items.begin(), items.end(),
  [=]( Item e )
{
  … your code here …
} );
```

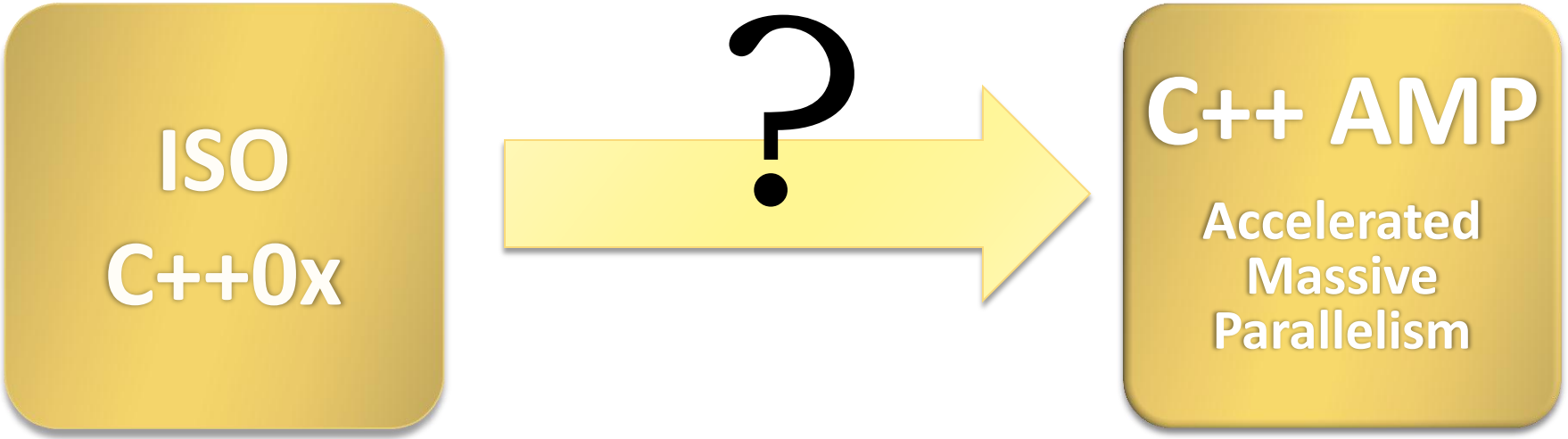
1

language feature for multicore

and STL, functors, callbacks, events, ...

Language Design: Parallelism Phase 2

Multi-core to hetero-core



Language Design: Parallelism Phase 2

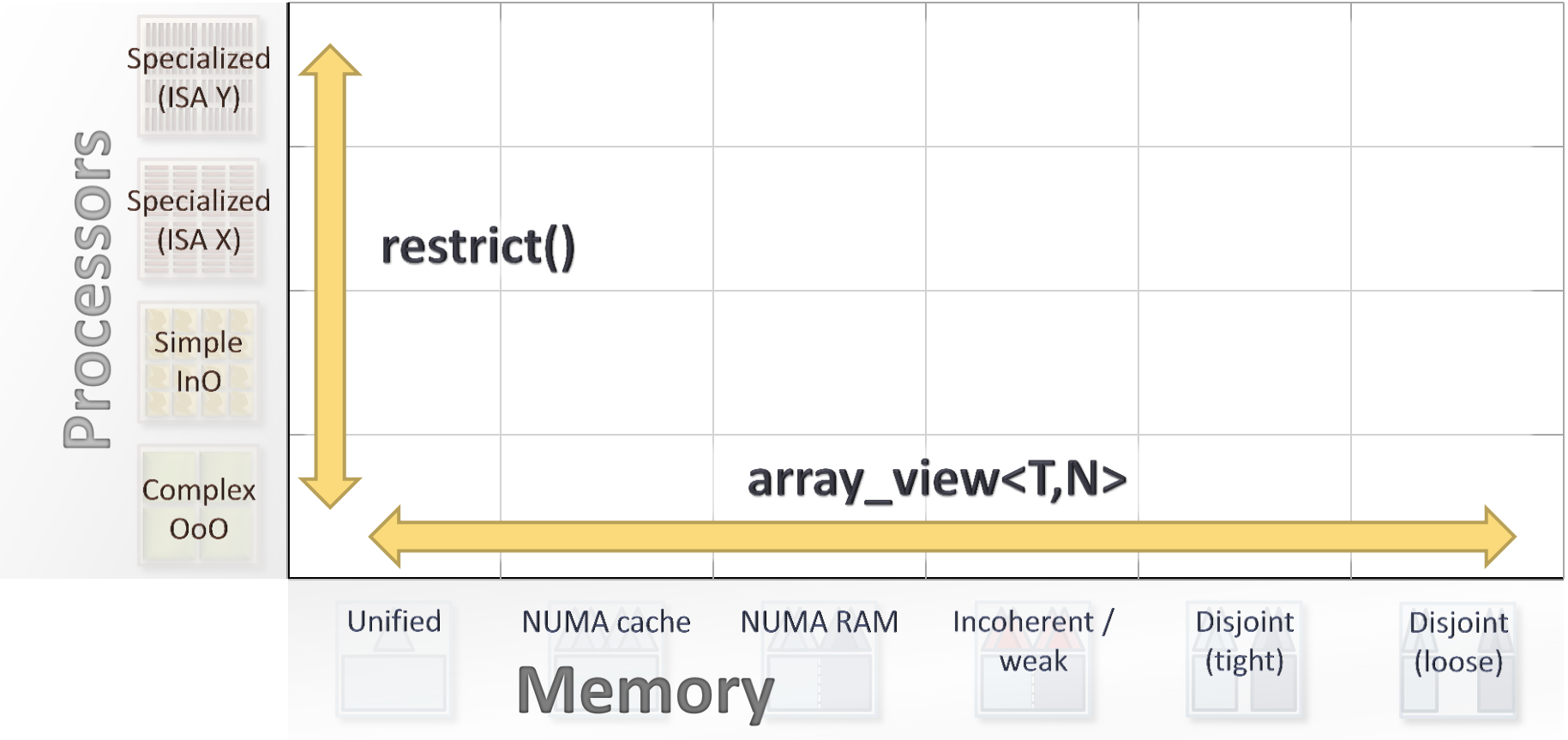
Multi-core to hetero-core


```
parallel_for_each(
   items.grid,
   [=](index<2> i) restrict(direct3d)
{
   … your code here …
} );
```

1

language feature for heterogeneous cores

C++ AMP at a Glance



restrict()

- **Problem:** Some cores don't support the entire C++ language.
- **Solution:** General restriction qualifiers enable expressing language subsets within the language. Direct3d math functions in the box.

Example

```
double sin( double );                          // 1a: general code
double sin( double ) restrict(direct3d);  // 1b: specific code

double cos( double ) restrict(direct3d); // 2: same code for either

parallel_for_each( c.grid, [=](index<2> idx) restrict(direct3d) {
   …
   sin( data.angle );  // ok, chooses overload based on context
   cos( data.angle );  // ok
   …
});
```

restrict()

- Initially supported restriction qualifiers:
  - **restrict(cpu)**: The implicit default.
  - **restrict(direct3d)**: Can execute on any DX11 device via DirectCompute.
    - Restrictions follow limitations of DX11 device model (e.g., no function pointers, virtual calls, goto).
- Potential future directions:
  - **restrict(pure)**: Declare and enforce a function has no side effects. Great to be able to state declaratively for parallelism.
  - General facility for language subsets, not just about compute targets.

array_view

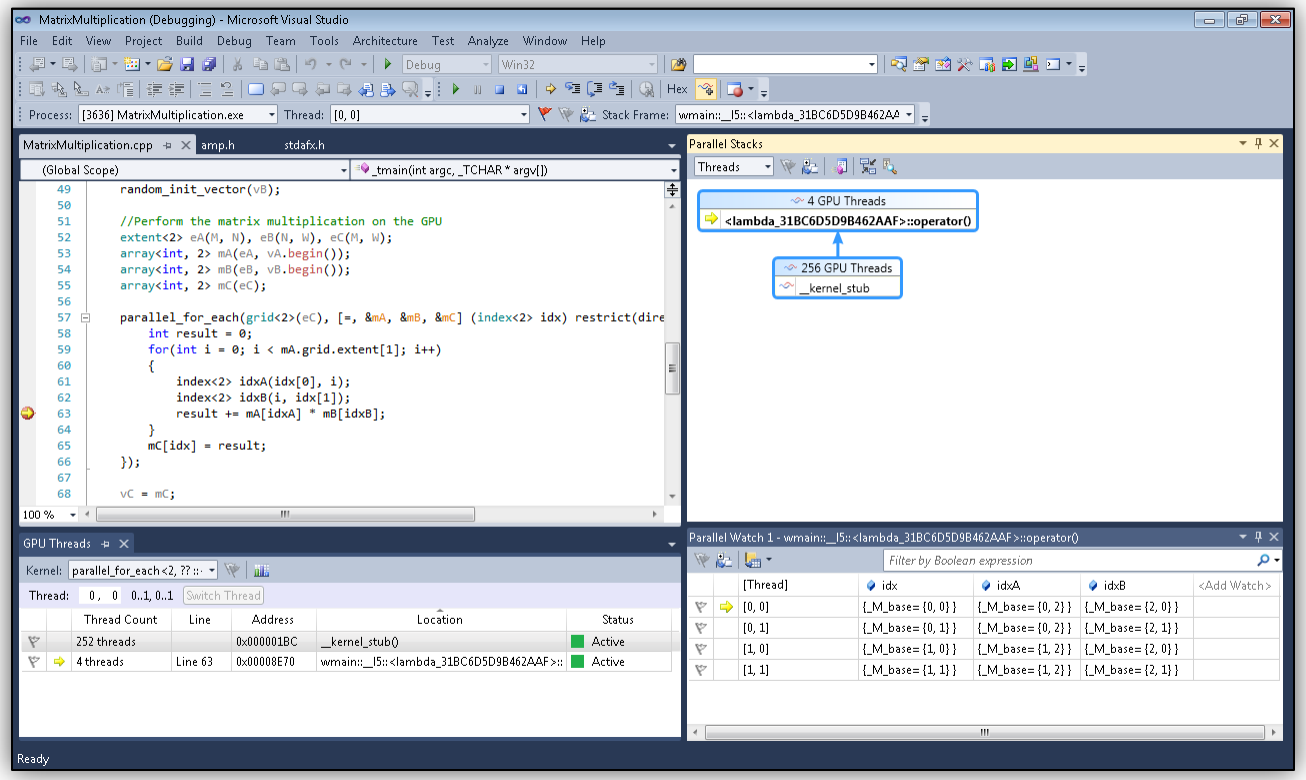
- **Problem:** Memory may be flat, nonuniform, incoherent, *and/or* disjoint.
- **Solution:** Portable view that works like an N-dimensional "iterator range."
  - Future-proof: No explicit .copy()/.sync(). As needed by each actual device.

Example

```
void MatrixMult( float* C, const vector<float>& A,
                 const vector<float>& B, int M, int N, int W )
{
  array_view<const float,2> a(M,W,A), b(W,N,B); // 2D view over C array
  array_view<writeonly<float>,2> c(M,N,C);      // 2D view over C++ std::vector

  parallel_for_each( c.grid, [=](index<2> idx) restrict(direct3d) {
    …
  } );
}
```

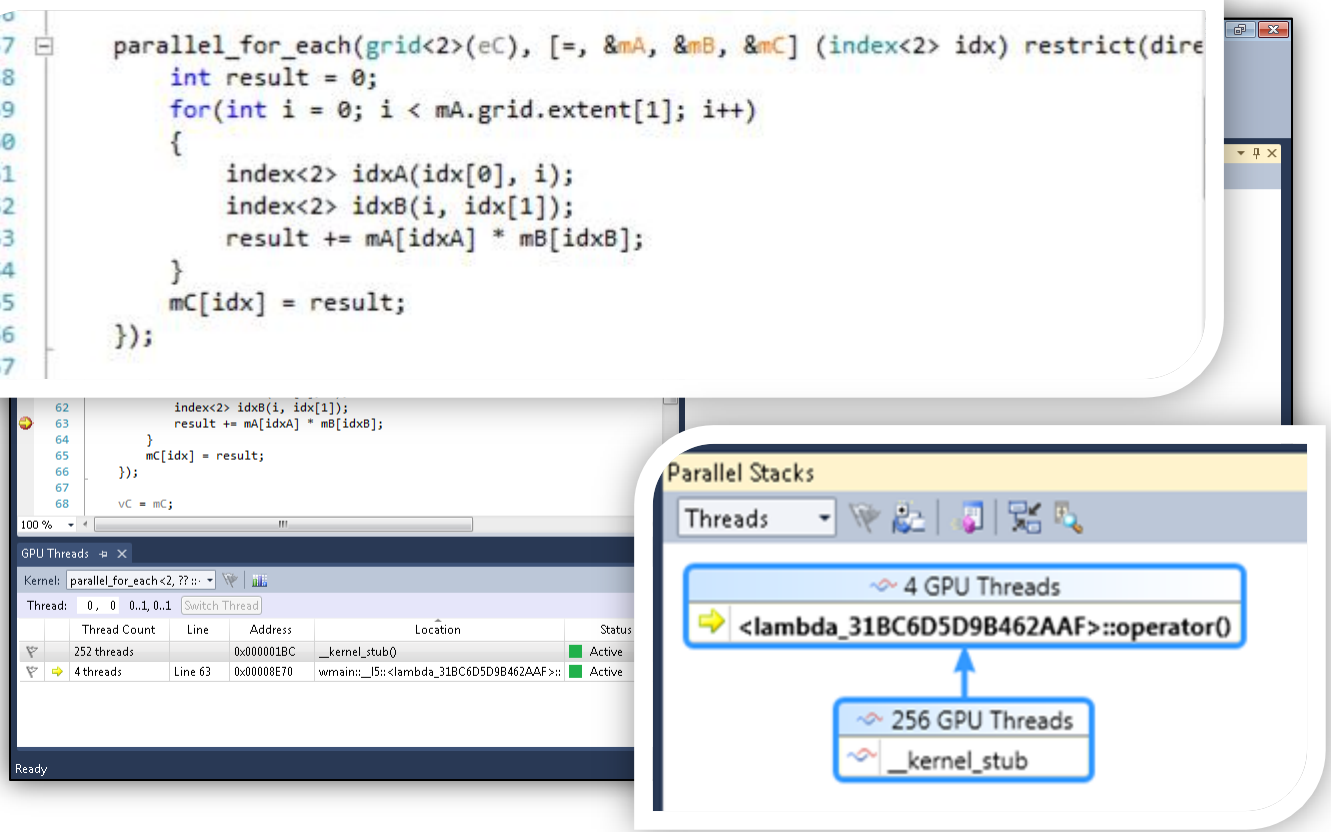
GPU Debugging



Bring CPU debugging experience to the GPU

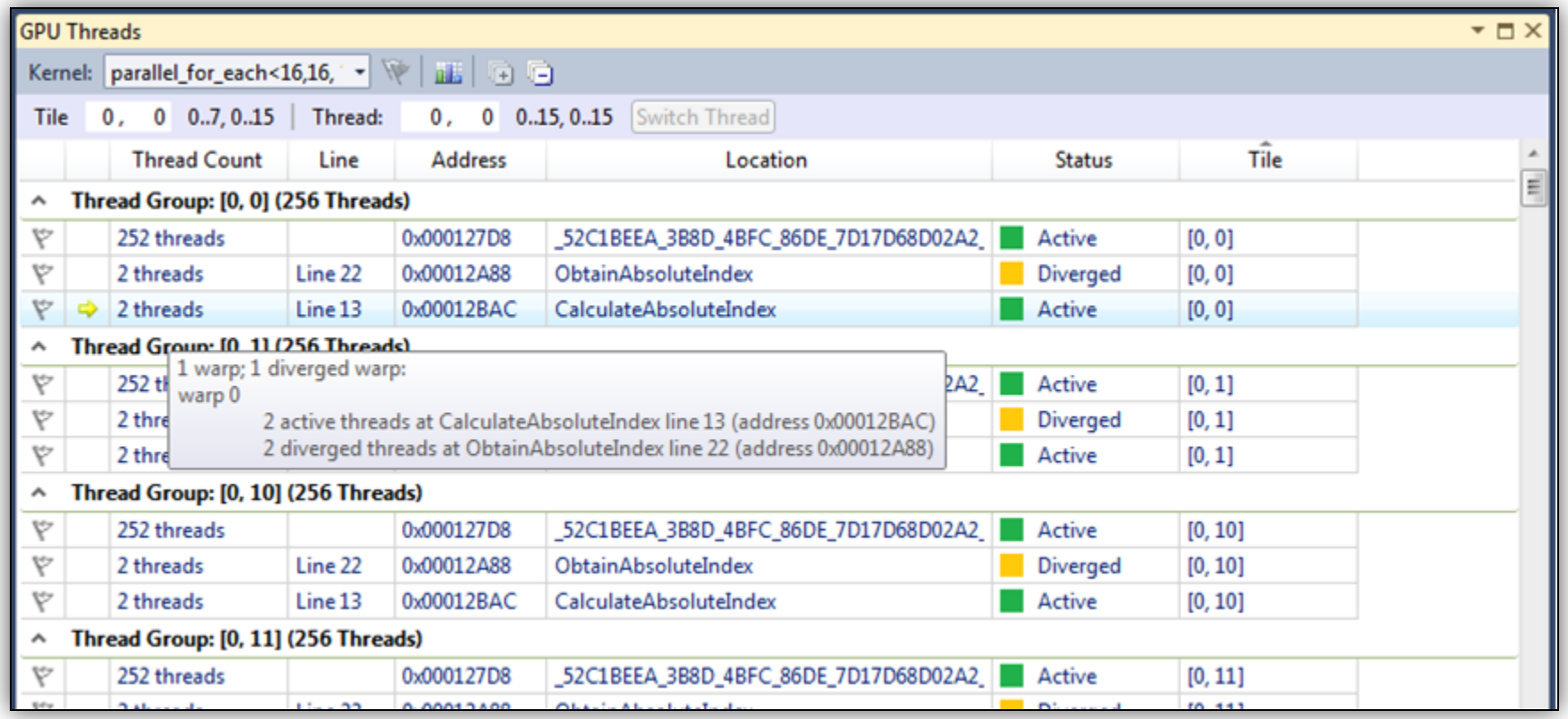
GPU Debugging

```
57    parallel_for_each(grid<2>(eC), [=, &mA, &mB, &mC] (index<2> idx) restrict(dire
58        int result = 0;
59        for(int i = 0; i < mA.grid.extent[1]; i++)
60        {
61            index<2> idxA(idx[0], i);
62            index<2> idxB(i, idx[1]);
63            result += mA[idxA] * mB[idxB];
64        }
65        mC[idx] = result;
66    });
67
```

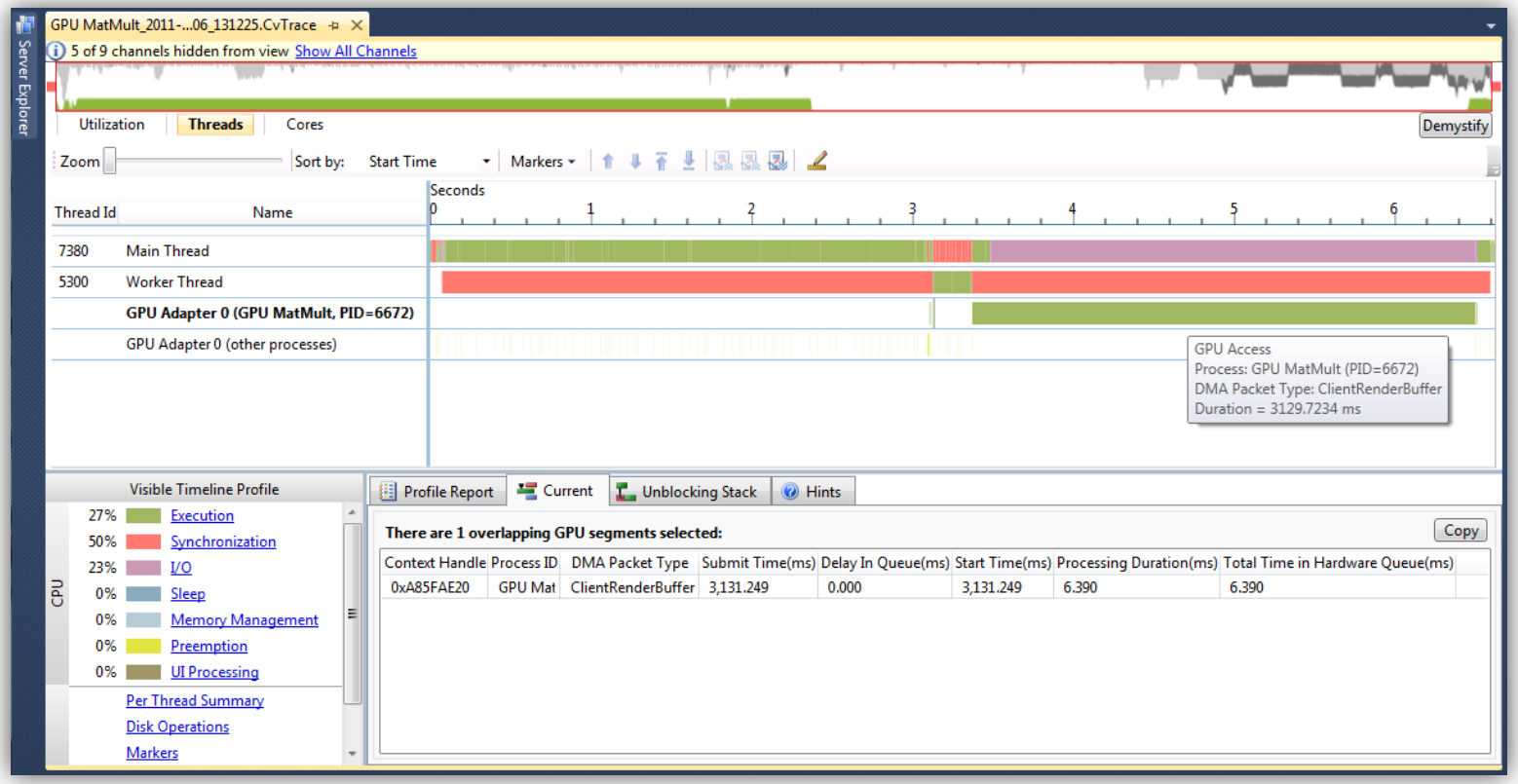


Bring CPU debugging experience to the GPU

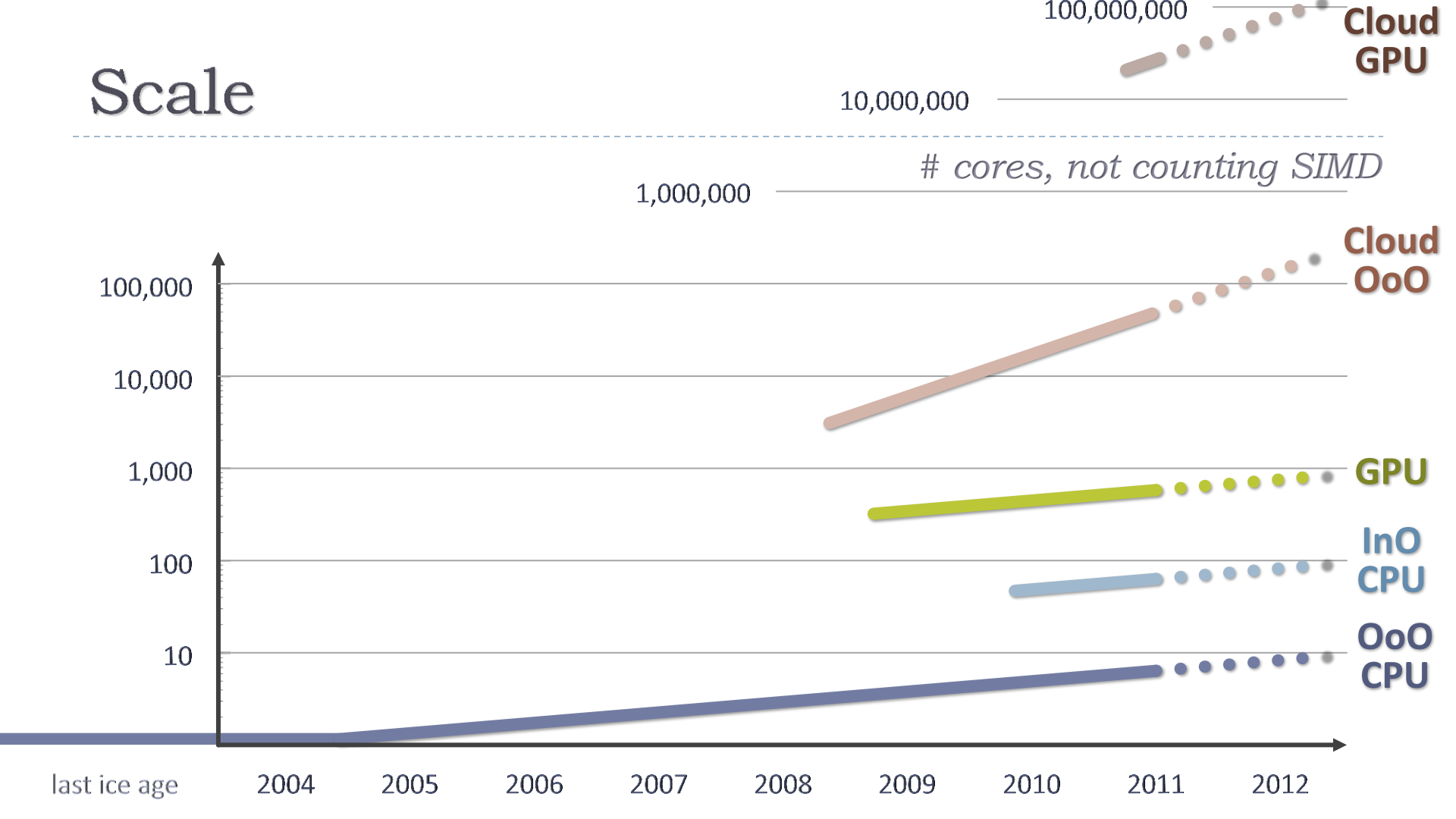
GPU Debugging



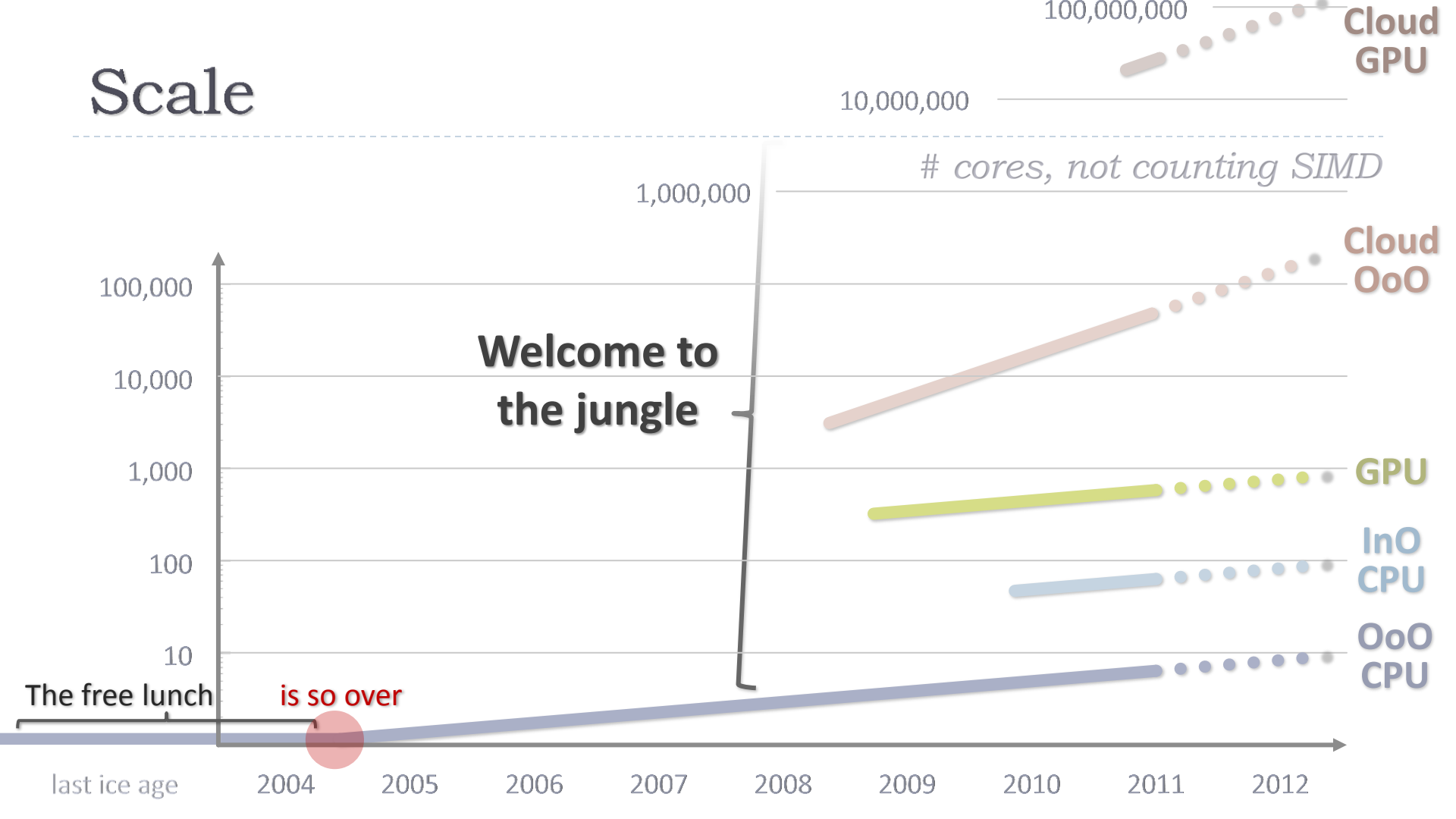
GPU Profiling



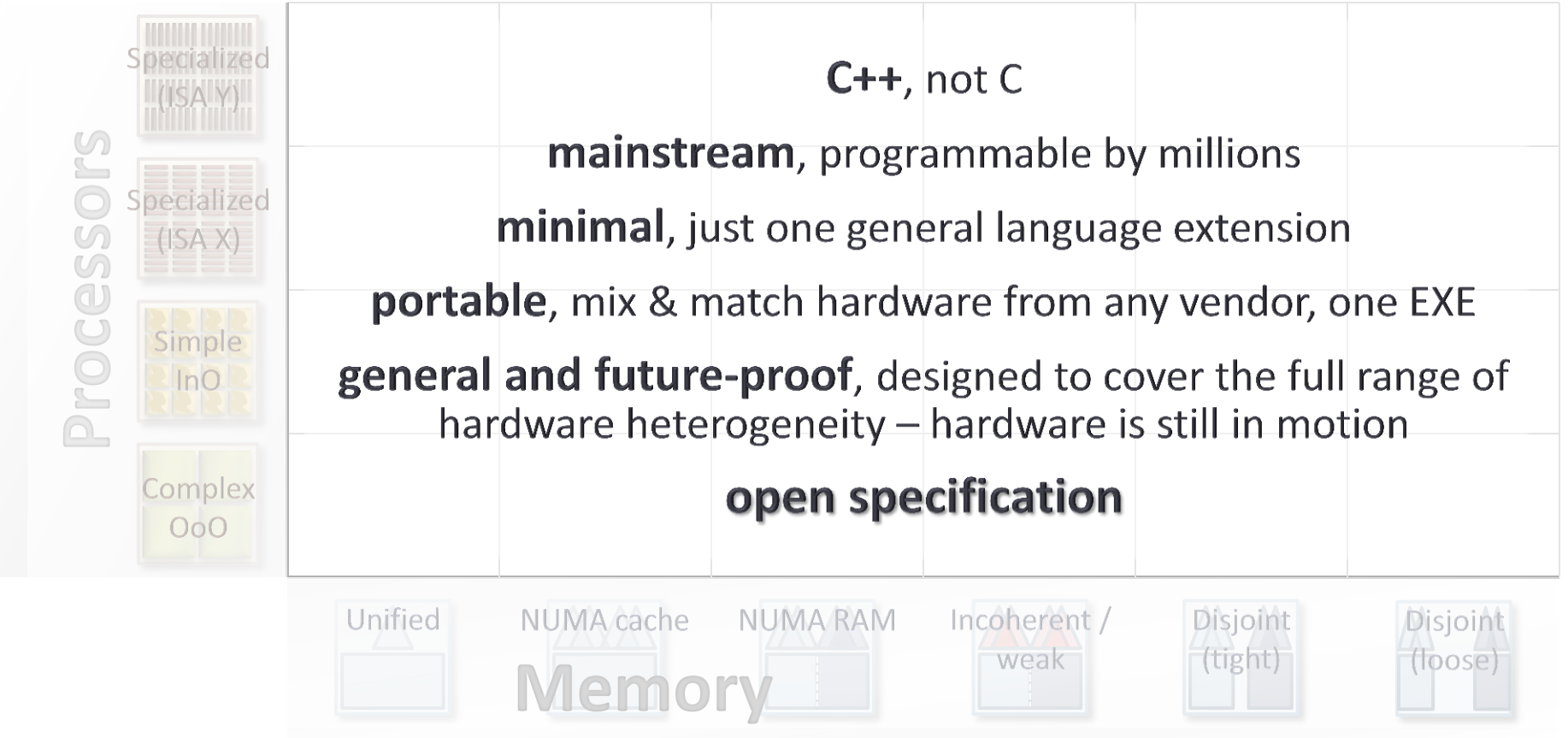
Scale


100,000,000
Cloud GPU
10,000,000
cores, not counting SIMD
1,000,000
Cloud OoO
100,000
10,000
1,000
GPU
100
InO CPU
10
OoO CPU
last ice age
2004
2005
2006
2007
2008
2009
2010
2011
2012

Scale


100,000,000
Cloud GPU
10,000,000
cores, not counting SIMD
1,000,000
Cloud OoO
100,000
10,000
Welcome to the jungle
1,000
GPU
100
InO CPU
10
OoO CPU
The free lunch
is so over
last ice age
2004
2005
2006
2007
2008
2009
2010
2011
2012

C++ AMP


Specialized (ISA Y)
Specialized (ISA X)
Simple InO
Complex OoO
Processors
C++, not C
mainstream, programmable by millions
minimal, just one general language extension
portable, mix & match hardware from any vendor, one EXE
general and future-proof, designed to cover the full range of hardware heterogeneity – hardware is still in motion
open specification
Unified
NUMA cache
NUMA RAM
Incoherent / weak
Disjoint (tight)
Disjoint (loose)
Memory

C++ PPL: 9:45am

C++ AMP: 2:00pm, Room 406

Heterogeneous Parallelism at Microsoft

Herb Sutter