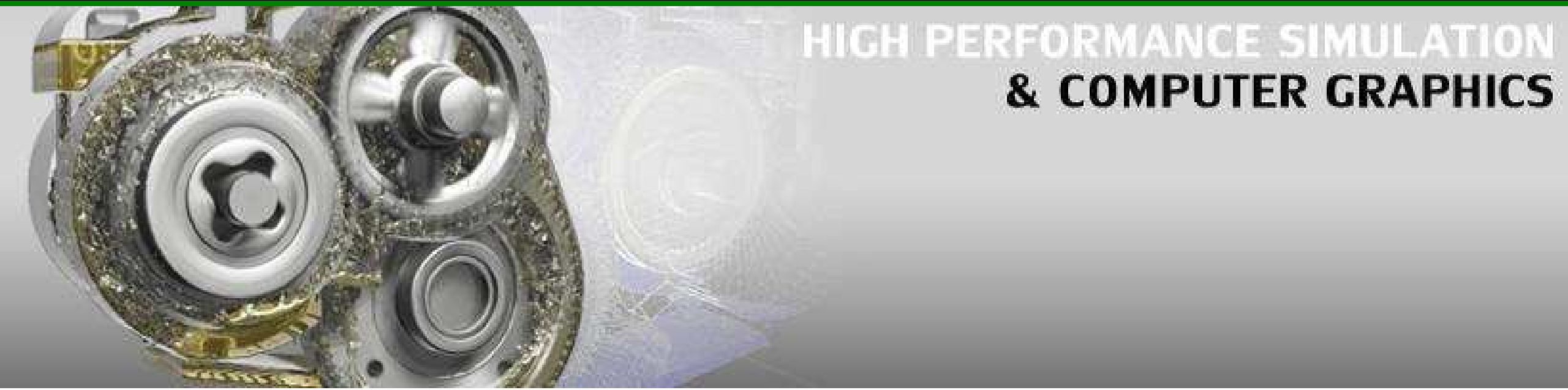


Particleworks: Particle-based CAE Software on Kepler

Yoshiaki Hanada

- Mar. 20, 2013 -

Prometech Software, Inc.

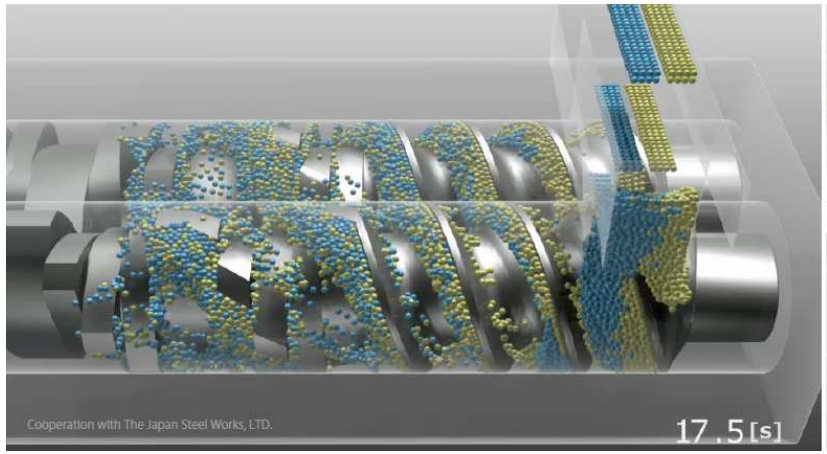
Agenda

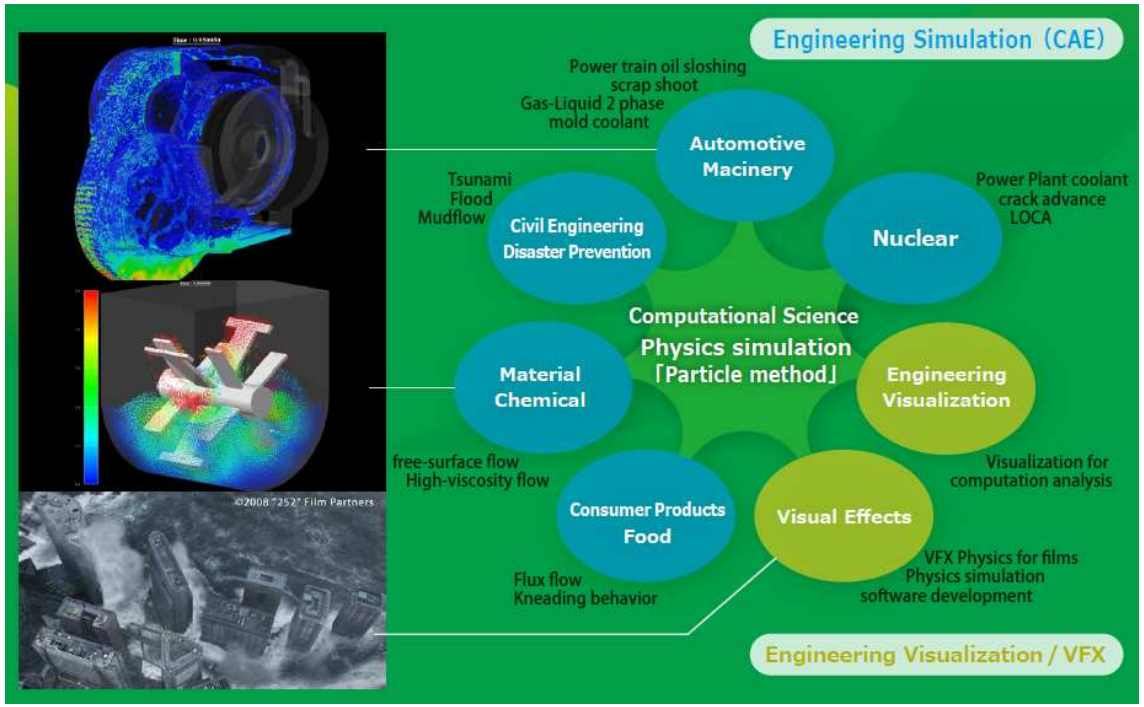
1. Company Profile
2. Particle-based CAE software “Particleworks”
3. Kepler optimization and performance
4. Summary

1. Company Profile

Prometech Software Inc.

- **Establishment** Oct. 29, 2004
- **Capital fund** $2.24M ($1=¥90)
- **Management team**
  Shinichi Okamoto (ex-CTO & VP at Sony Computer Entertainment Inc.),
  Toshimitsu Fujisawa, Ph.D (Founder), Seiichi Koshizuka, Ph.D (Founder),
  Yoshiaki Hanada, Akihiro Sawaii, Tatsuo Yuguchi
- **Business overview**
  Software development and sales about particle-based Computer Aided Engineering
- **Major shareholders**
  KOZO KEIKAKU Engineering Inc.(4748 JQS)
  Mitsubishi UFJ Capital Co., Ltd.
  Founders, collaborative researchers and employees, etc.
- **Location**
  The University of Tokyo Entrepreneur Plaza 3F,
  Hongo 7-3-1, Bunkyo-ku, Tokyo, JAPAN





1. Company Profile – Particle simulation

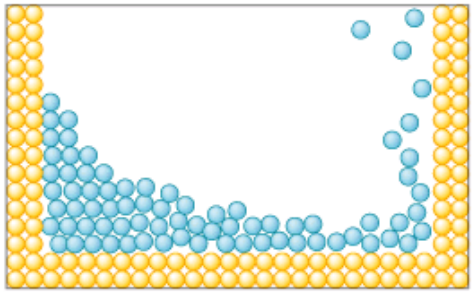
MPS method (Moving Particle Simulation) was invented by Prof. Seiichi Koshizuka to simulate severe accidents at a nuclear power plant in 1990's. And now it is widely used in Japanese various industries.

THE UNIVERSITY OF TOKYO

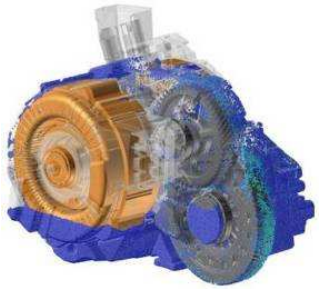
Prof. Seiichi Koshizuka

Inventor of MPS method and Founder of Prometech Software

- School of Engineering, the University of Tokyo

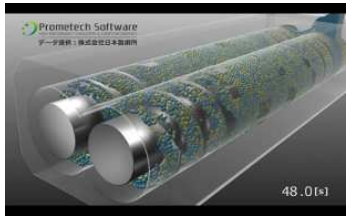


Auto, Machinery



Oil behavior in HV transaxle
Courtesy of TOYOTA MOTOR CORPORATION.

Material



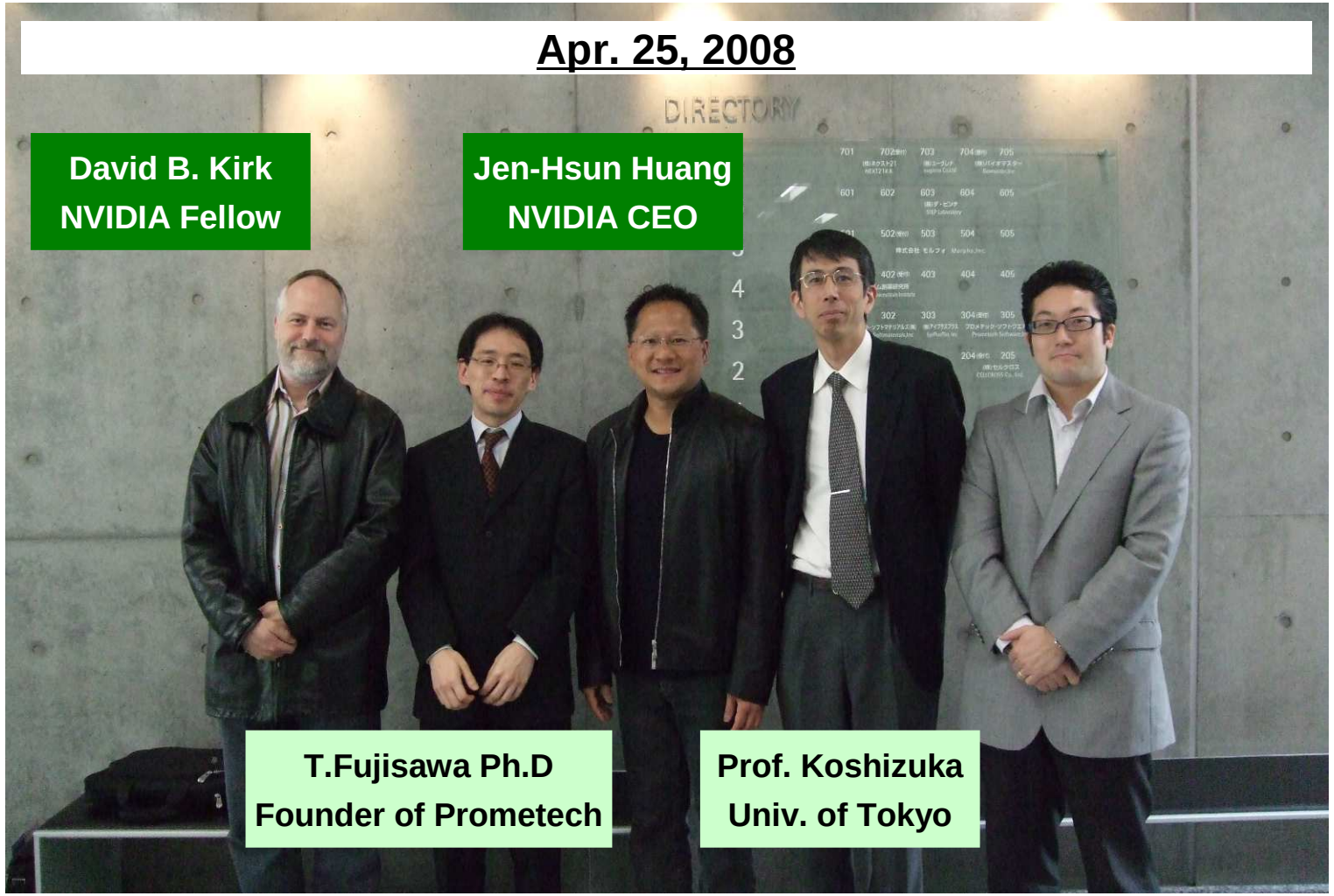
Twin screw extruder analysis
Courtesy of The Japan Steel Works, LTD.

Nuclear, Civil engineering

Mudslide and mudslide-control dam simulation, River flood simulation, etc.

1. Company Profile – GPU challenge

Since Jen-Hsun Huang's visit to Prometech Software in 2008, we have been developing GPU ready software and maximizing our business opportunities with NVIDIA's innovative GPU technology.



2010

- Particleworks v2.0
  - ✓Matrix solver was ported on GPU

2011

- Particleworks v2.5
  - ✓Fully ported on GPU
- Particleworks v3.0
  - ✓New GUI
  - ✓Powder simulation (DEM) solver on CPU was released

2012

- Particleworks v4.0
  - ✓Fluid (MPS) and Powder (DEM) solvers were ported on GPU

2013

- Particleworks v4.5
  - ✓Ready for Kepler GPU

Agenda

1. Company Profile
2. Particle-based CAE software “Particleworks”
3. Kepler optimization and performance
4. Summary

2. Particle-based CAE software “Particleworks”

Particle works™

Particle-based simulation software for CAE

DESIGNED FOR NVIDIA CUDA

[Main features]

- Free-surface / non-steady flow (MPS) simulation
- Fluid - Powder (MPS-DEM) coupling simulation
- Fluid - Rigid body coupling simulation
- Non-Newtonian flow simulation
- Turbulence model
- Negative pressure model
- Surface tension model
- Air resistance model
- Pressure calculation options (Explicit / Implicit)
- Viscosity calculation options (Explicit / Implicit)
- Parallel calculation options (SMP / MPP / Hybrid)
- GPU calculation, etc…

2. Particle-based CAE software “Particleworks” – Type of particle simulation

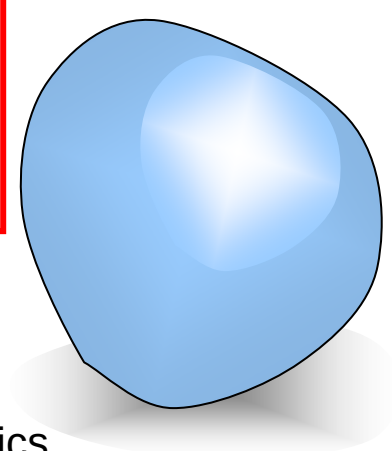
There are some types of particle simulation. In Particleworks, MPS is used for fluid and DEM is used for powder.

Fluid

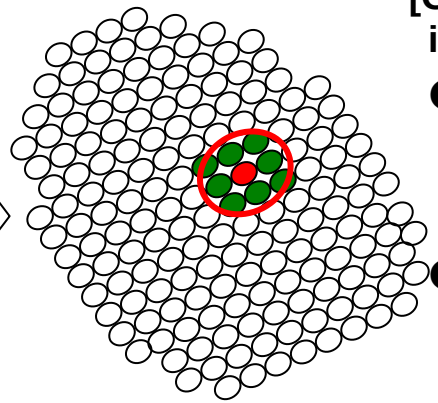
Describe continuum body as a group of particles, and calculate physical quantities which are distributed on each particles.
(See continuum body as a group of particles)

MPS*
Moving
Particle
Simulation

SPH
Smoothed
Particle
Hydro dynamics



Continuum body



Group of particles

[Governing eq. of incompressible flow]

- Navier-Stokes eq.

$$\frac{D\vec{u}}{Dt} = -\frac{\nabla P}{\rho} + \nu\nabla^2\vec{u} + g$$

- Eq. of continuity

$$\frac{D\rho}{Dt} = 0$$

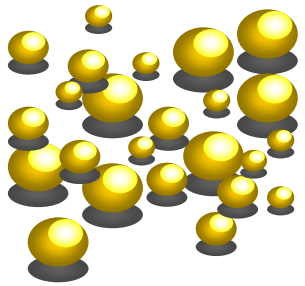
Solve governing eq. of each particles

Update positions of each particles

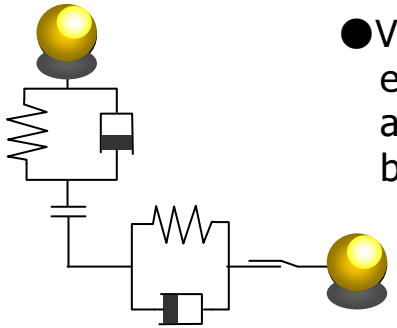
Powder

Use springs and dash-pots to describe all forces, and calculate eq. of motion for each particles.

DEM
Discrete
Element
Method



Discrete elements



Voigt model

- Voigt model shows elastic forces by springs and viscous damping by dash-pots

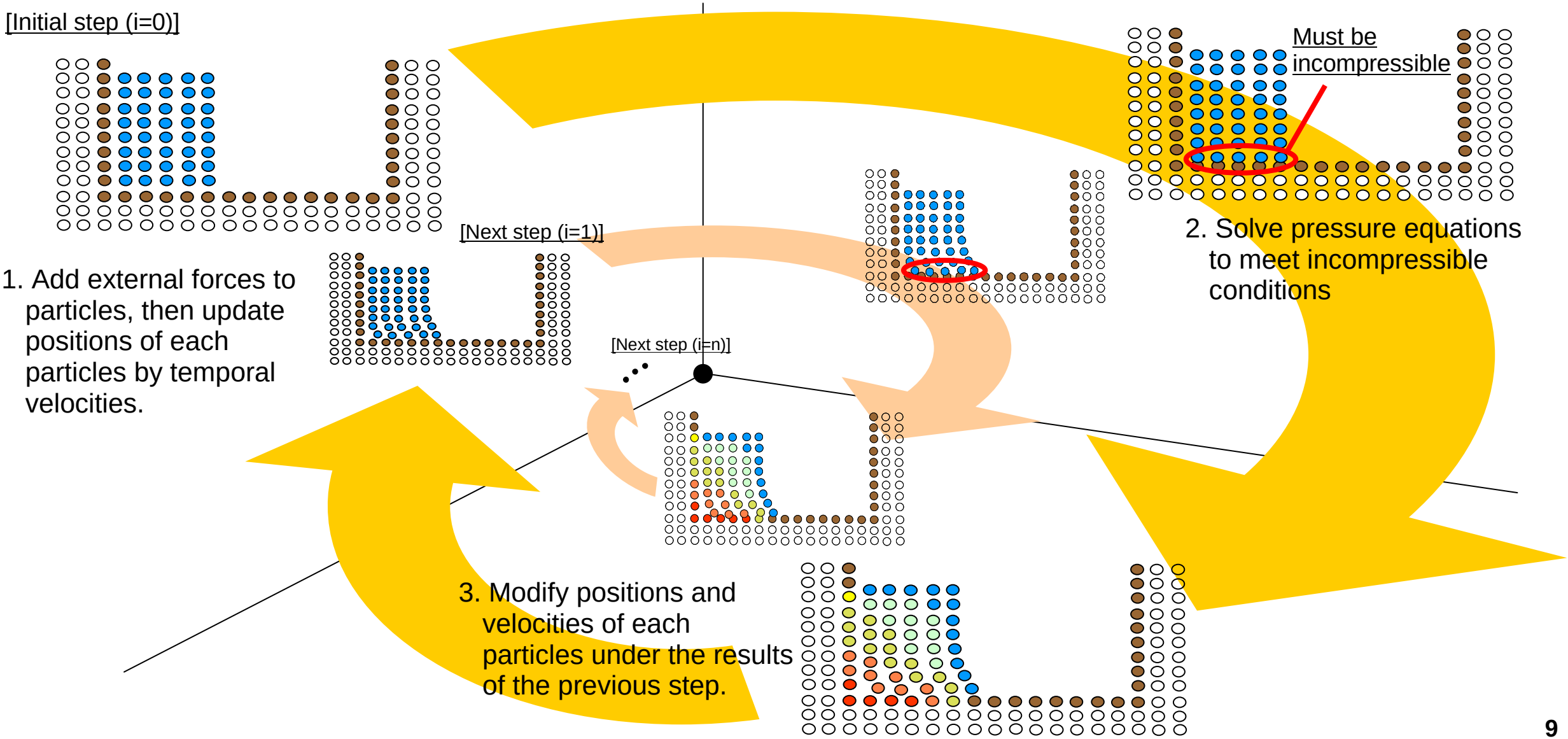
Solve eq. of motion of each particles

Update positions of each particles

(*) Moving-Particle Semi-Implicit Method for Fragmentation of Incompressible Fluid, NUCLEAR SCIENCE AND ENGINEERIGN 123 421-434 (1996), S.Koshizuka and Y.Oka

2. Particle-based CAE software “Particleworks” – Calculation procedure

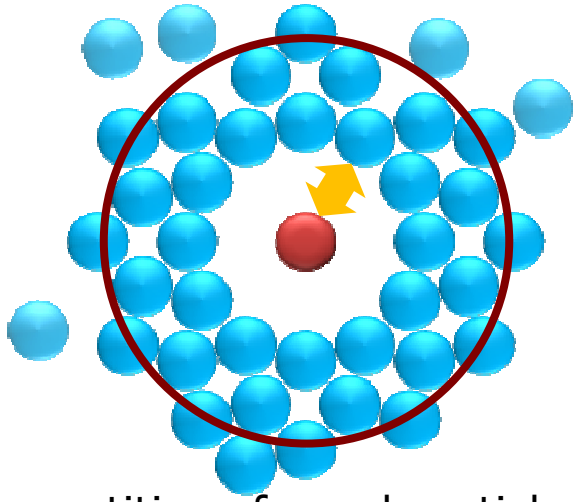
For example, a simple fractional time stepping is used as one of time marching methods in Particleworks.



2. Particle-based CAE software “Particleworks” – Performance bottleneck

In the calculation procedure of MPS, FLOPS and random memory access affect calculation performance.

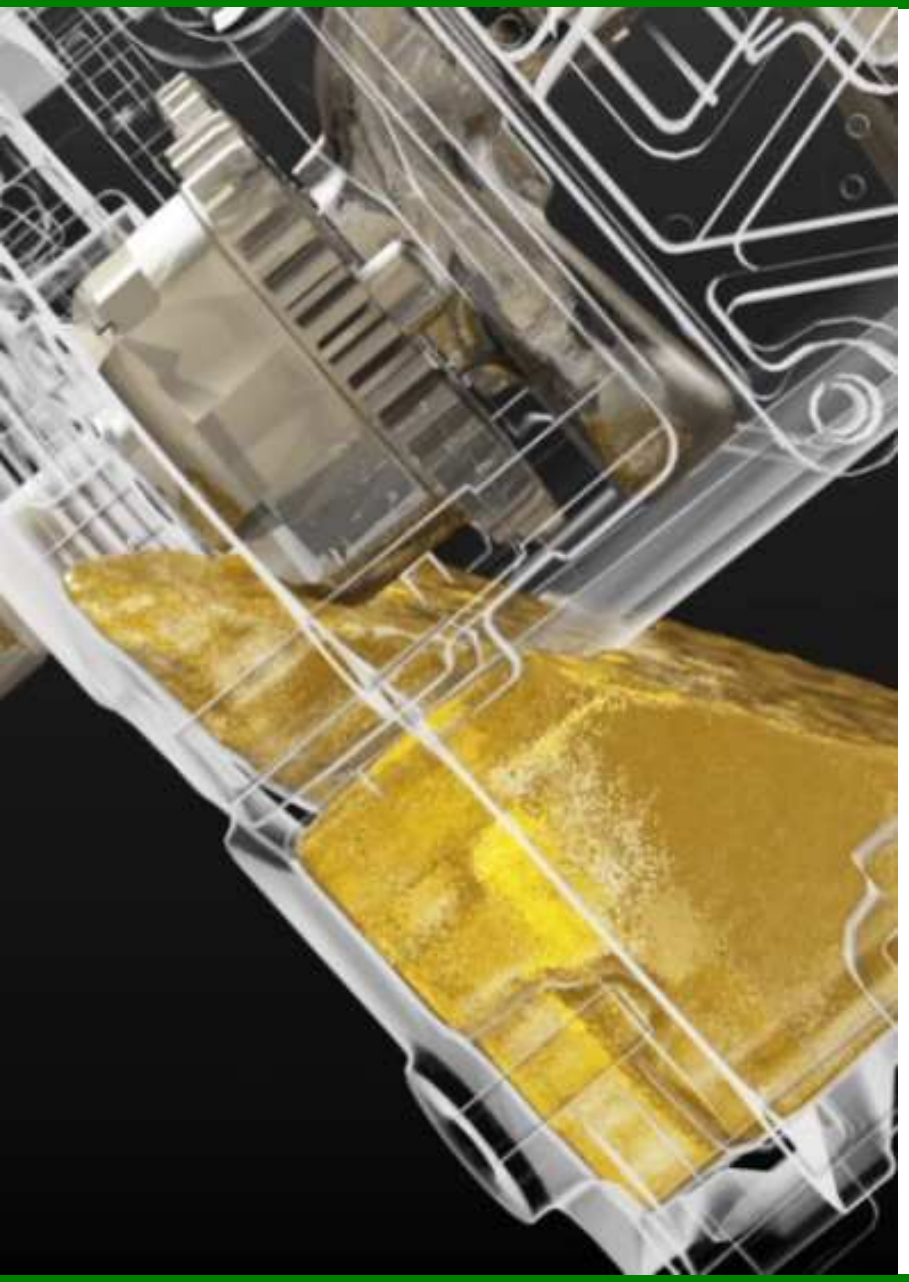
1. Add external forces, then update physical quantities.
2. Solve pressure equations to meet incompressible conditions
3. Modify positions and velocities in accordance with above results



All physical quantities of a red particle are decided by interactions between the red particle and blue particles within the effective radius.

Performance bottleneck	**Optimization ideas**
FLOPS	Instruction throughput enhancement
Random memory access	Hide access latency

Agenda

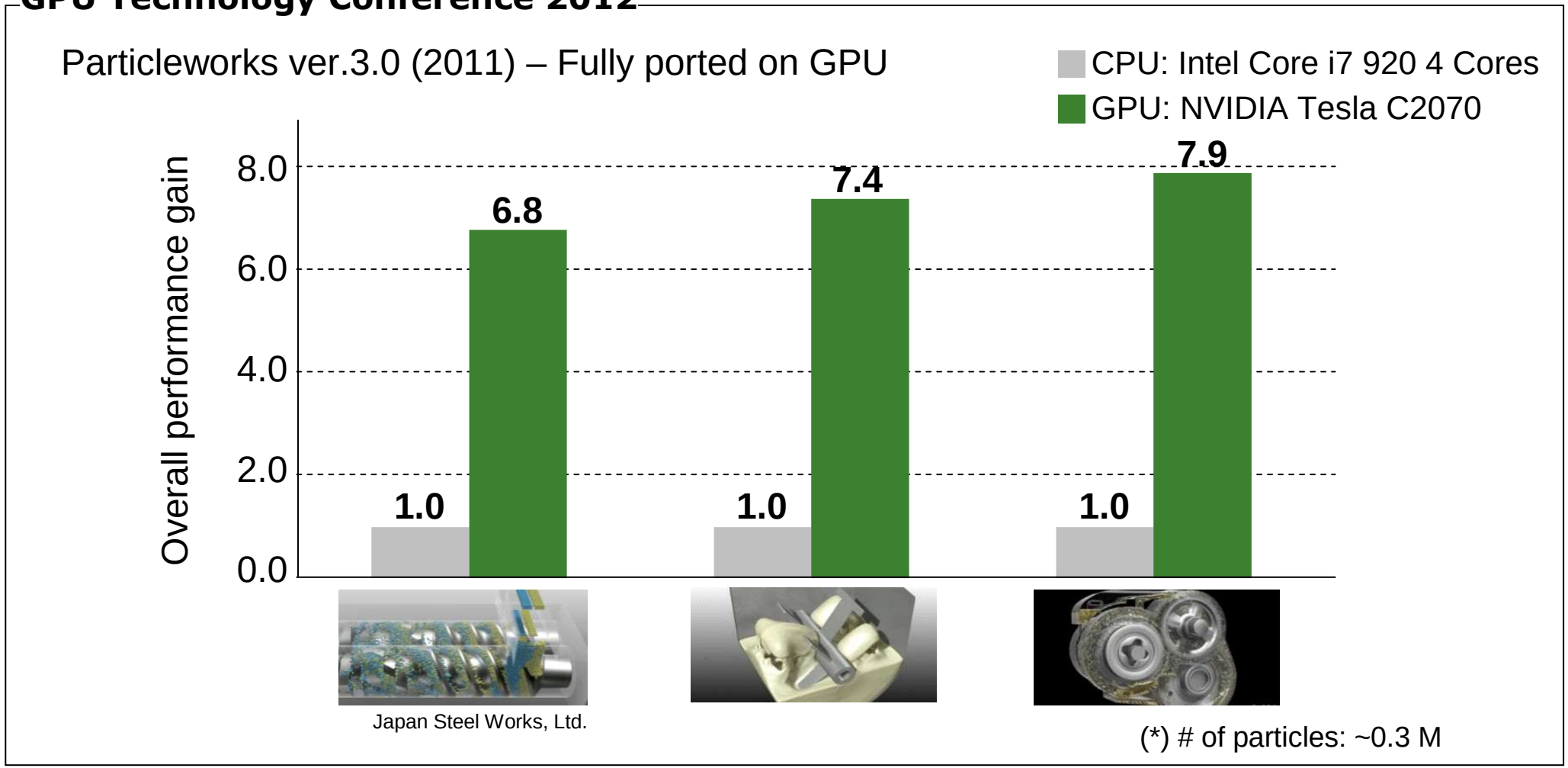


1. Company Profile
2. Particle-based CAE software “Particleworks”
3. Kepler optimization and performance
4. Summary

3. Kepler optimization and performance – Past achievement

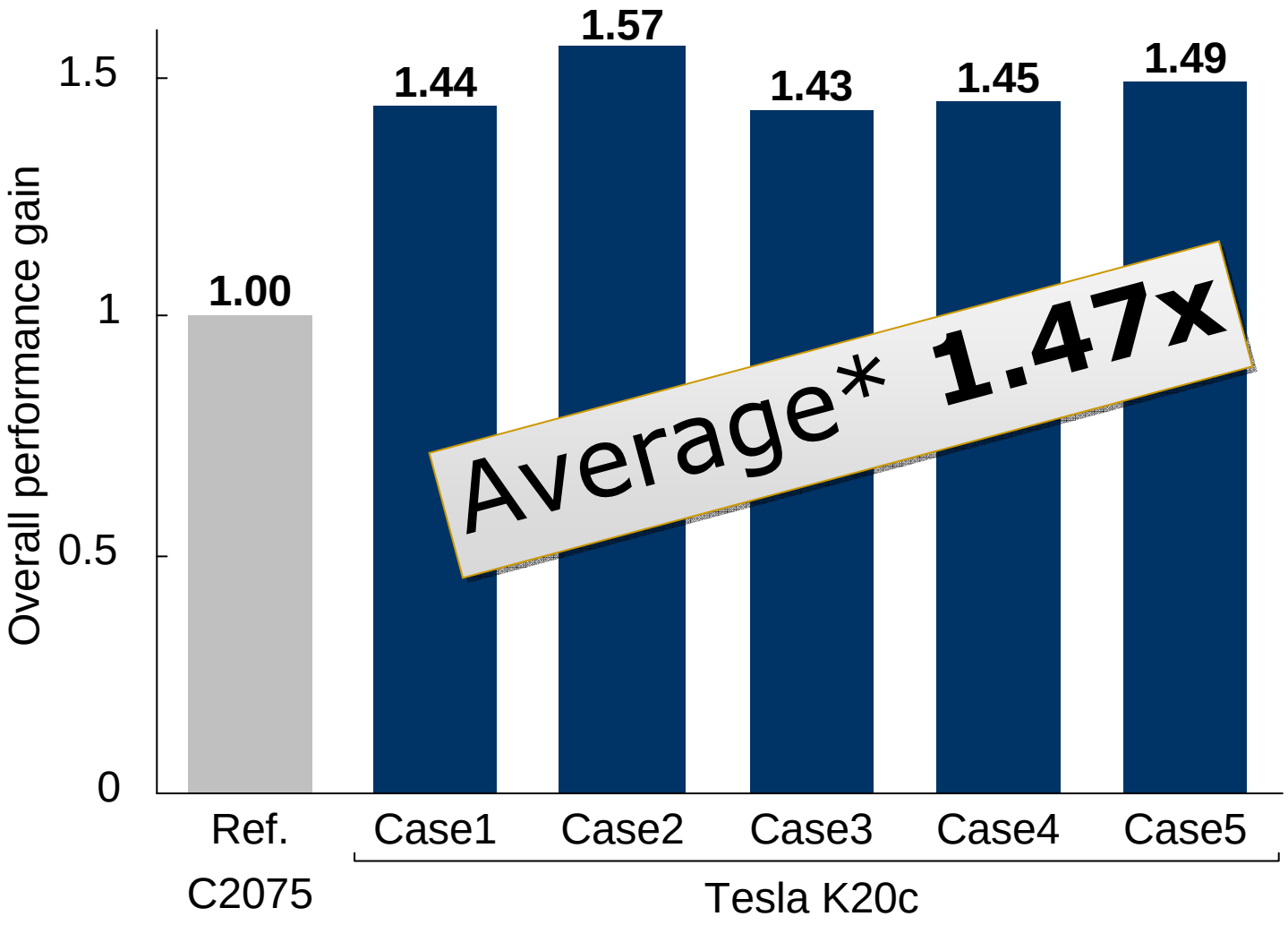
Last GTC, Prometech reported that particle simulation which is fully ported on GPU (C2075) showed ~7x performance gain compared to Core i7 4 core CPU.

GPU Technology Conference 2012



3. Kepler optimization and performance – From C2075 to K20c

Prometech measured Particleworks performance with 5 different cases, and Particleworks on K20c is 1.47x (average) faster than Particleworks on C2075



[Case1]

High-viscosity mixing

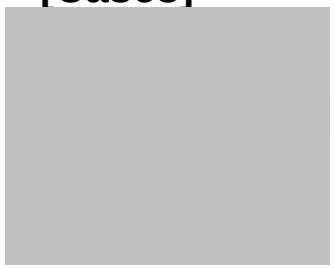
Courtesy of Mitsubishi Chemical Corporation.

[Case2]

Barrel finishing machine analysis

Courtesy of Tipton Corp

[Case3]



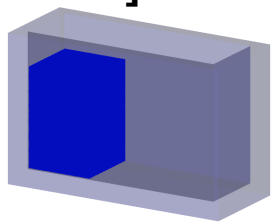
Balancer in laundry machine analysis

Courtesy of a home appliance company.

[Case4]

Gearbox oil behavior analysis

[Case5]



Dam break analysis

(*) Geometric mean

3. Kepler optimization and performance – Performance summary and question

Performance gains are roughly uniform on average.

Case Number	Case 1	Case 2	Case 3	Case 4	Case 5
# of Particles	807,885	344,633	366,210	295,113	861,042
Pressure	Implicit	Implicit	Implicit	Implicit	Implicit
Viscosity	Implicit	Explicit	Explicit	Implicit	Explicit
Surface Tension		x		x	
Turbulence				x	
DEM		x			
Performance Gain	**1.44**	**1.57**	**1.43**	**1.45**	**1.49**

3. Kepler optimization and performance – Tesla specifications

Double precision performance of K20c is 2.05x higher (measured by nbody.exe), and memory bandwidth performance is 1.36x faster (measured by bandwidthTest.exe) than Tesla C2075.

	GPU	Tesla C2075 (Fermi)		Tesla K20c (Kepler)
Calculation performance	GPU Clock Rate	1.15 GHz		0.71 GHz
	# of CUDA Cores	448 = (14) MPs x (32) Cores/MP		2,496 = (13) MPs x (192) Cores/MP
	Double-precision (*1)	~238 GFLOPS/s	**2.05x**	~489 GFLOPS/s
Memory performance	Memory Clock Rate	1.57 GHz		2.60 GHz
	Memory Bas Width	384-bit		320-bit
	Memory Bandwidth (*2)	~119 GB/s	**1.36x**	~162 GB/s

(*1) measured by nbody.exe
(*2) measured by bandwidthTest.exe

3. Kepler optimization and performance – Concept of optimization (1/8)

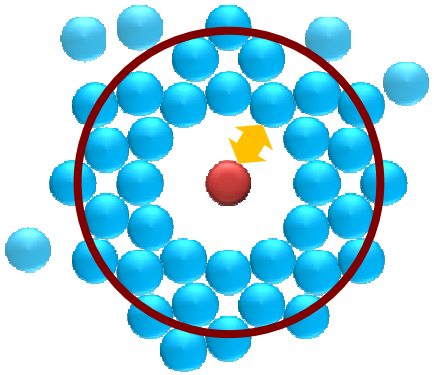
SpMV is a frequently occurred operation in CAE program. It is the same in particle simulation that SpMV is used for calculating interactions between particles.

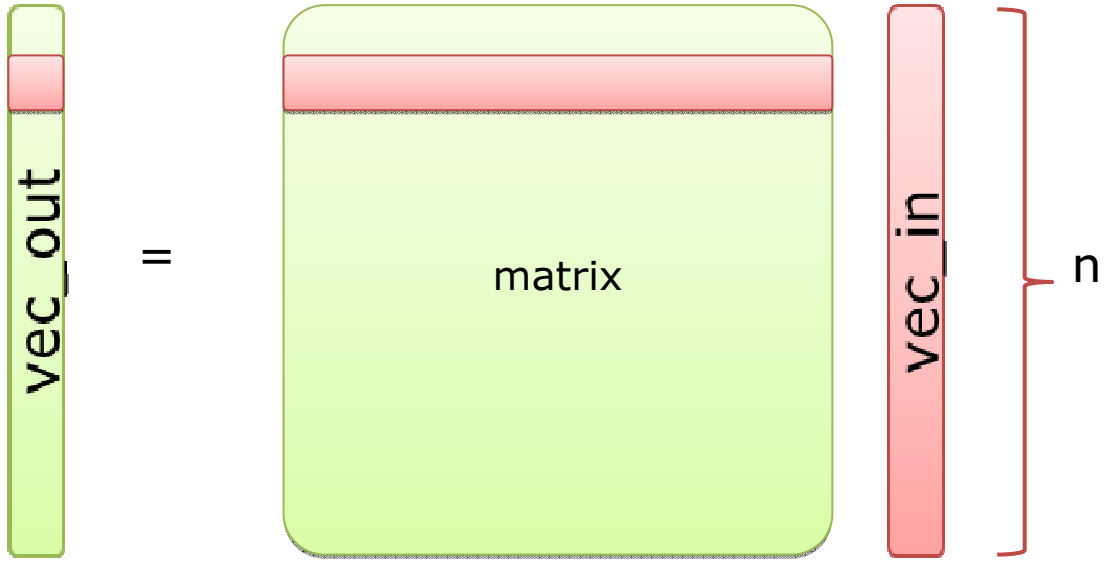
Sparse Matrix-Vector Multiplication (CSR format) on CPU (C code)

```
for (int i = 0; i < n; ++i) {
  double sum = 0.0;
  for (int p = row_ptr[i], end = row_ptr[i+1]; p < end; ++p) {
    sum += val[p] * vec_in[col_ind[p]]
  }
  vec_out[i] = sum;
}
```

- Number of elements in vector:
  n = 100K, (100K dimension vector)
- Number of non-zero elements in matrix: 10M
- Double-precision floating-point numbers

* SpMV is well used in implicit (pressure / viscosity) calculation.
  In particle simulation, this formula describes the interactions between particles.





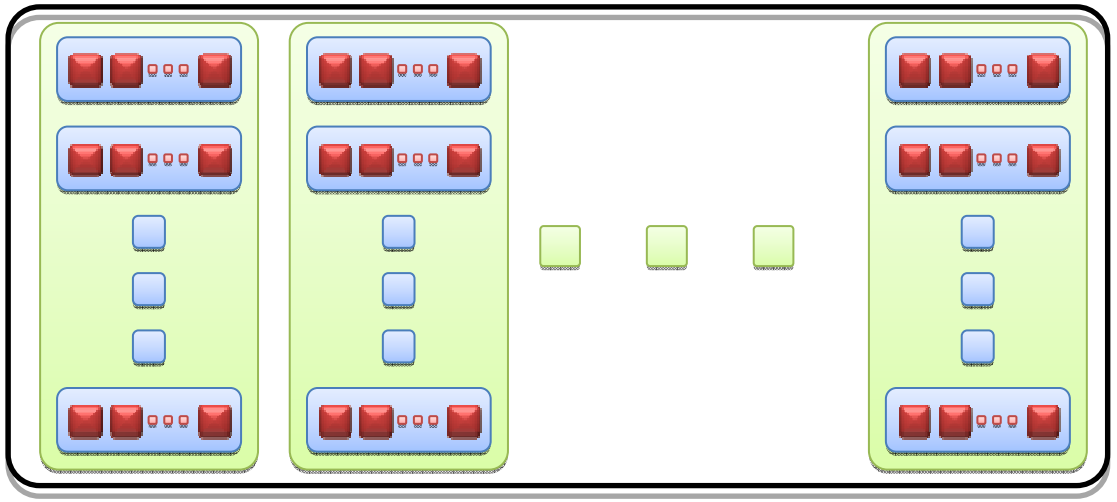
3. Kepler optimization and performance – Concept of optimization (2/8)

GPU is a hierarchical hardware, and we always care about “warp” when we develop Particleworks.

Hierarchy in GPU

- Device (GPU)
  - Block (SM / SMX)
    - Warp
      - Thread (CUDA Core)

GPU architecture image



- warp is a minimum executable unit
  - ✓ Warp is a high functional SPMD compared to SIMD
- It is necessary to write processing by warp to achieve good effective performance.

3. Kepler optimization and performance – Concept of optimization (3/8)

For example, SpMV on Fermi can be written as follows.

Sparse Matrix-Vector Multiplication on Fermi (C2075)

```
__global__ void spmv_kernel(
    int                                 const n,
    int             const       *       const row_ptr,
    int             const       *       const col_ind,
    double          const       *       const val,

    double                      *       const vec_out)
{
  extern __shared__ double smem[];

  int const tid = threadIdx.x;
  int const lid = tid & 31;
  int const stride = (blockDim.x * gridDim.x) >> 5;
  for (int i = (tid + blockIdx.x * blockDim.x) >> 5; i < n; i += stride) {
    double sum = 0.0;
    for (int p = row_ptr[i] + lid, end = row_ptr[i+1]; p < end; p += 32) {
      int2 const v = tex1Dfetch(tex_vec_in, col_ind[p]);
      sum += val[p] * __hiloint2double(v.y, v.x);
    }
    smem[tid] = sum; sum += smem[tid+16];
    smem[tid] = sum; sum += smem[tid+8];
    smem[tid] = sum; sum += smem[tid+4];
    smem[tid] = sum; sum += smem[tid+2];
    smem[tid] = sum; sum += smem[tid+1];
    if (lid == 0) {
      vec_out[i] = sum;
    }
  }
}
```

Use texture memory for random memory access

Parallel reduction

Execute 1 line of a matrix in 1 warp

3. Kepler optimization and performance – Concept of optimization (4/8)

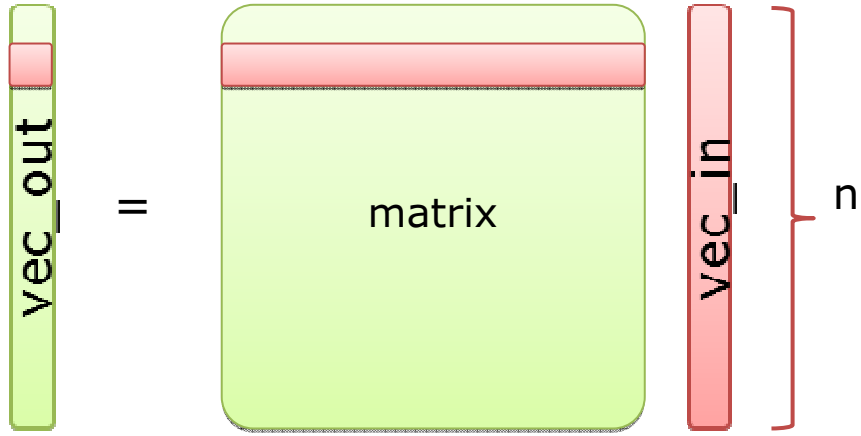
When the SpMV program which is for Fermi is executed on Kepler without any change, the performance is 1.23x and this result is inferior to the hardware memory bandwidth improvement.

SpMV (executed on Tesla C2075 and Tesla K20c)

- Number of elements in vector: n = 100K, (100K dimension vector)
- Number of non-zero elements in matrix: 10M
- Double-precision floating-point numbers

Result (measured by NVIDIA Visual Profiler)

- Performance gain is just 1.23x and this is inferior to the memory bandwidth improvement.
- We need optimization for Kepler



GPU code	Fermi Version on Tesla C2075		Fermi Version on Tesla K20c	Kepler Version on Tesla K20c
GPU time (us)	1,544	1.23x	1,252	---

Memory bandwidth improvement, but inferior to hardware spec.

(Ref.) see P15.

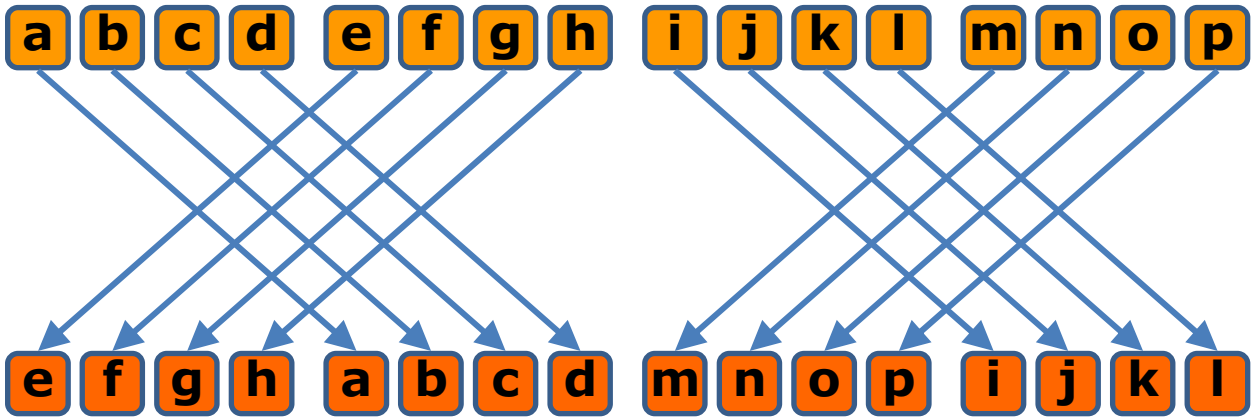
GPU	Tesla C2075 (Fermi)		Tesla K20c (Kepler)
Memory Bandwidth	~119 GB/s	1.36x	~162 GB/s

3. Kepler optimization and performance – Concept of optimization (5/8)

Kepler implements a new Shuffle instruction, which allows threads within a warp to share data. A few percent performance gain can be seen just by using Shuffle.

Shuffle instruction within a warp

- ✓ Example) __shfl_xor(value, 4)



New function from Compute Capability 3.x (Kepler)

- ✓ This allows threads within a warp to share data
- ✓ In the case of FFT, which requires to data sharing within a warp, a 6% performance gain can be seen just by using Shuffle.
- ✓ For more information, see NVIDIA's whitepaper, "NVIDIA's Next Generation CUDA™ Compute Architecture: Kepler ™ GK110"

3. Kepler optimization and performance – Concept of optimization (6/8)

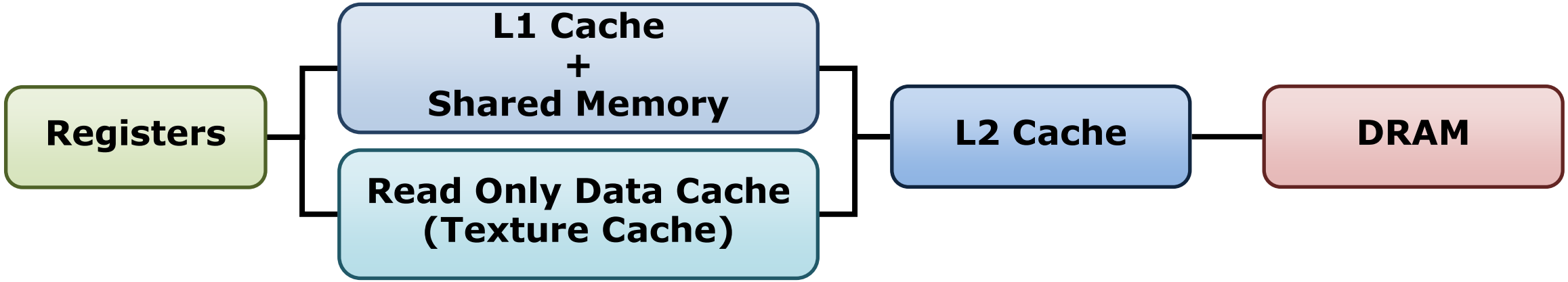
In addition to the L1 cache, Kepler introduces a 48KB cache for data that is known to be read-only for the duration of the function. Use of the read-only path is beneficial.

Use of the read-only path is beneficial

- ✓ Because it takes both load and working set footprint off of the Shared/L1 cache path.
- ✓ In addition, the Read-Only Data Cache's higher tag bandwidth supports full speed unaligned memory access patterns among other scenarios.

Use of this path is managed automatically by the compiler

- ✓ Any variable or data structure which "const __restrict" keyword will be tagged will be loaded through the Read-Only Data Chache.



- ✓ Choose variables or data structure carefully not to disturb cache lines to use "const __restrict".

3. Kepler optimization and performance – Concept of optimization (7/8)

As a result, SpMV on Kepler is as follows.

Sparse Matrix-Vector Multiplication on Kepler (K20c)

```
__global__ void spmv_kernel(
    int                                 const n,
    int       const       *             const row_ptr,
    int       const       *             const col_ind,
    double    const       *             const val,
    double    const       * __restrict  const vec_in,
    double                *             const vec_out)
{


  int const tid = threadIdx.x;
  int const lid = tid & 31;
  int const stride = (blockDim.x * gridDim.x) >> 5;
  for (int i = (tid + blockIdx.x * blockDim.x) >> 5; i < n; i += stride) {
    double sum = 0.0;
    for (int p = row_ptr[i] + lid, end = row_ptr[i+1]; p < end; p += 32) {

      sum += val[p] * vec_in[col_ind[p]];
    }
    sum += shfl_xor(sum, 16);
    sum += shfl_xor(sum,  8);
    sum += shfl_xor(sum,  4);
    sum += shfl_xor(sum,  2);
    sum += shfl_xor(sum,  1);
    if (lid == 0) {
      vec_out[i] = sum;
    }
  }
}
```

✓Use const * __restrict instead of texture memory

```
__device__ __inline__
  double shfl_xor(double const value, int const laneID)
{
  return __hiloint2double(
              __shfl_xor(__double2hiint(value), laneID),
              __shfl_xor(__double2loint(value), laneID));
}
```

✓Use Warp Shuffle Operation

3. Kepler optimization and performance – Concept of optimization (8/8)

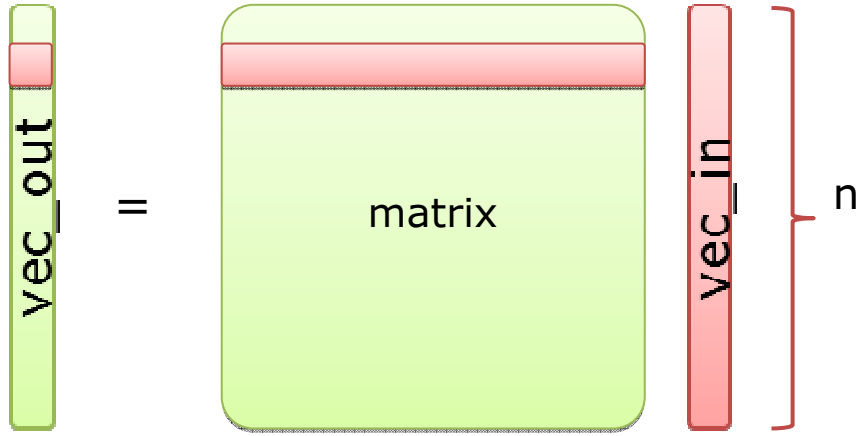
Performance gain is 1.48x compared to the original version.

SpMV (executed on Tesla C2075 and Tesla K20c)

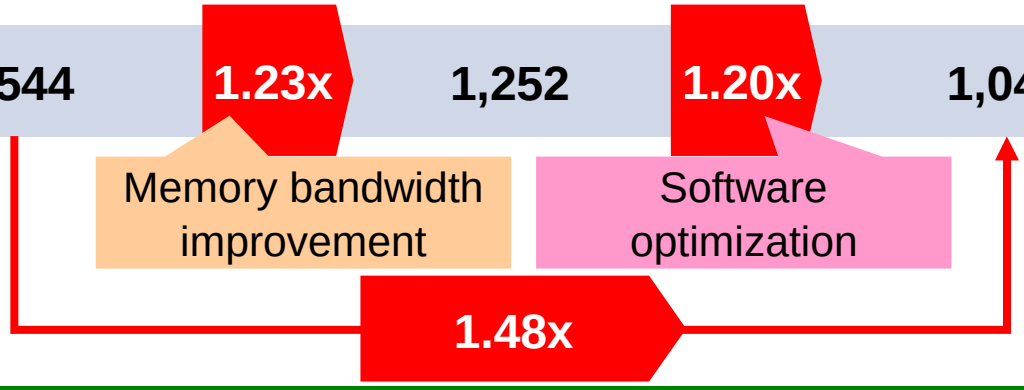
- Number of elements in vector: n = 100K, (100K dimension vector)
- Number of non-zero elements in matrix: 10M
- Double-precision floating-point numbers

Result (measured by NVIDIA Visual Profiler)

- Performance gain is 1.48x and this is superior to the memory bandwidth improvement.
- It is important for a programmer to develop a program which use memory bandwidth effeciently.



GPU code	Fermi Version on Tesla C2075	Fermi Version on Tesla K20c	Kepler Version on Tesla K20c
GPU time (us)	1,544	1,252	1,043



Agenda

1. Company Profile
2. Particle-based CAE software “Particleworks”
3. Kepler optimization and performance
4. Summary

4. Summary

- Prometech Software Inc. is a particle-based CAE software start-up in Japan.
- Particleworks is particle-based commercial CAE software and widely used in Japan.
- In Particleworks, 1.47x performance gain can be seen on K20c. (compared to C2075)
- In the case of SpMV operation which is a frequently occurred calculation in particle simulation, Prometech showed some optimization ideas from Fermi to Kepler.
  - ✓ CUDA code for Fermi can't show enough performance gain on Kepler. (It falls below the memory bandwidth improvement of hardware.)
  - ✓ Prometech uses warp shuffle instruction and read-only data cache to improve performance.
  - ✓ As a result, it shows 1.48x performance gain in total.


High Performance Simulation & Computer Graphics
Prometech Software