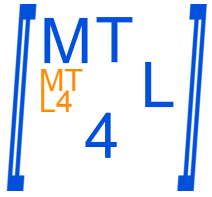
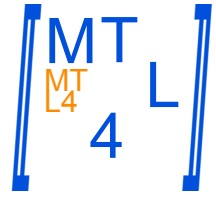


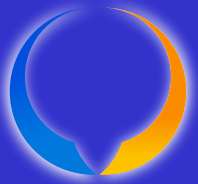
SimuNova



Cleverer Programmieren im Prozessordschungel

Peter Gottschling

SimuNova



1. Motivation
   - Software-Entwicklung
   - Ansatz: Generische Programmierung
2. MTL4: Intuitiv Programmieren
   - Matrixoperationen
   - Löser
3. Zu neuen Ufern: zukunftssichere Anwendungen
   - Portabilität
   - PMTL4: Massiv-parallel
   - CUDA-MTL4: GPUs
   - Performance
4. Software-Ökosystem
   - Debugging
   - Nutzer

Motivation

- Hardware immer diverser
  - Multi-Core, Many-Core,
  - GPU,
  - Cell (RIP),
  - Cluster von all denen
- Verschiedene Speicherstrukturen
  - Hierarchisch auf jeden Fall,
  - Extra Speicher für GPUs,
  - Eventuell verteilt
- Algorithmen und Daten nicht unbedingt homogener

Motivation

- Hardware immer diverser
  - Multi-Core, Many-Core,
  - GPU,
  - Cell (RIP),
  - Cluster von all denen
- Verschiedene Speicherstrukturen
  - Hierarchisch auf jeden Fall,
  - Extra Speicher für GPUs,
  - Eventuell verteilt
- Algorithmen und Daten nicht unbedingt homogener
- Zukünftige Entwicklung wird nicht einfacher

Softwarekomplexität reduzieren

- Algorithmen und Datenstrukturen entkoppeln
- Algorithmen und Hardwareaspekte entkoppeln
  - Parallelität abstrahieren
  - Datenlokalität in speziellen Modulen verwalten, nicht in Anwendungen
- Intuitive Schnittstellen
- Wissenschaftler sollen sich nicht mit niederen technischen Details herumschlagen
- Sondern Modelle und Algorithmen entwickeln

Summierung auf Feldern

```
int sum(int *array, int n)
{
  int s = 0;
  for (int i = 0; i < n; ++i)
    s = s + array[i];
  return s;
}
```

```
double sum(double *array, int n)
{
  double s = 0;
  for (int i = 0; i < n; ++i)
    s = s + array[i];
  return s;
}
```

Das sollte cleverer gehn.

Templatisierte Summe auf Feldern

```
template <typename T>
T sum(T *array, int n)
{
  T s = 0;
  for (int i = 0; i < n; ++i)
    s = s + array[i];
  return s;
}
```

Einführung von Templates.

Summierung verschiedener Container

```
int sum(int *array, int n)
{
  int s = 0;
  for (int i = 0; i < n; ++i)
    s = s + array[i];
  return s;
}
```

```
double sum(node *first)
{
  double s = 0;
  while (first) {
    s = s + first->data;
    first = first->next;
  }
  return s;
}
```

Können wir das vereinheitlichen?

Summierung mit Iteratorbereichen

Einführung des Input-Iterators — jeder Typ mit:

- Inkrementoperator;
- Dereferenzierung und
- Vergleich.

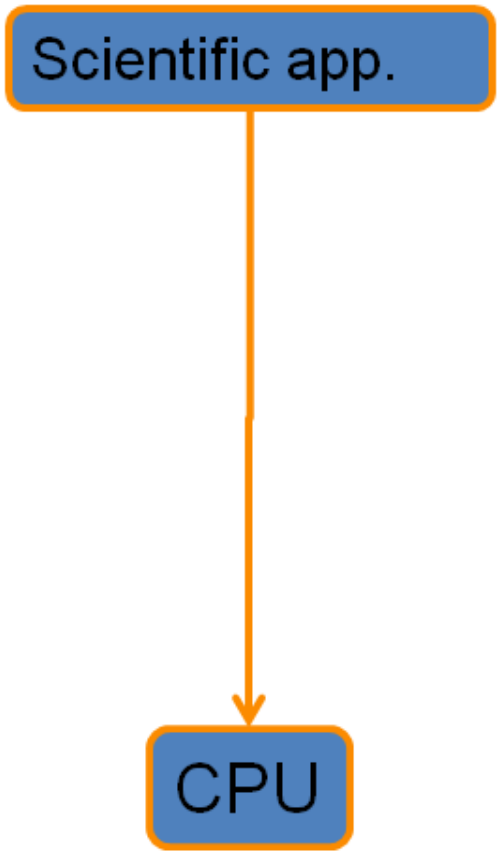
Schreibe Algorithmen für rechtsoffene Intervalle davon:

```
template <typename Iter, typename T>
T sum(Iter first, Iter last, T s)
{
  for ( ; first != last; ++first)
    s = s + *first;
  return s;
}
```

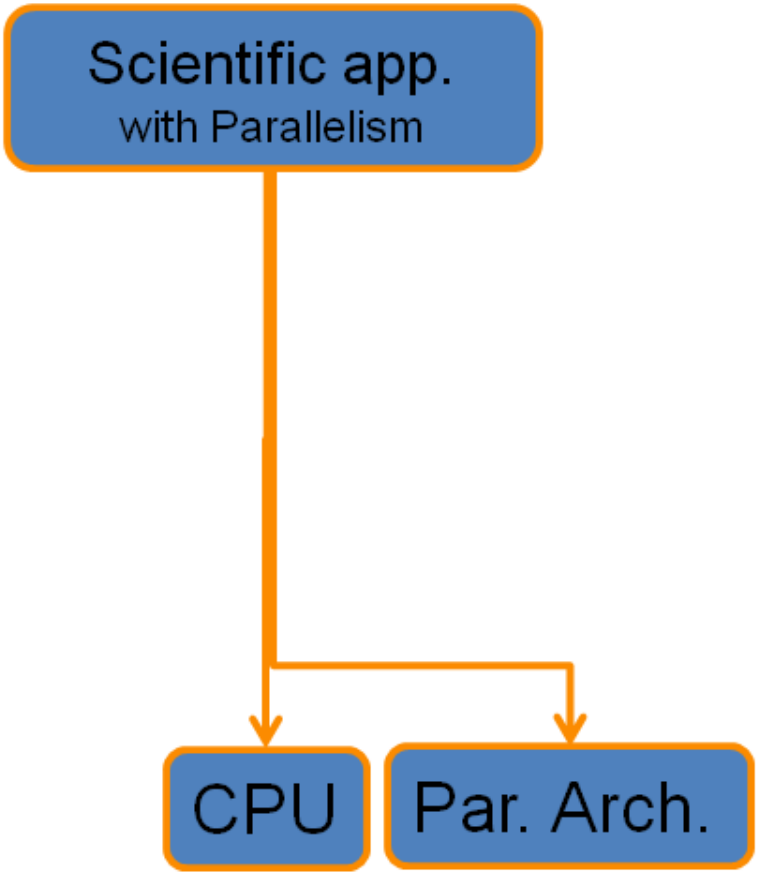
```
int main(int, char**)
{
    mtl::dense2D<double> A, B= {{1, 2}, {3, 4}};
    mtl::matrix<double, morton> C= {{3, 2}, {7, 5}};
    mtl::matrix<float, sparse, col_major> D= {{2, 0}, {0, 3}};

    // Entspricht A= B * B; aus Nostalgie; lieber nicht nutzen
    mult(B, B, A);

    A= B * B; // Nimm BLAS (fuer grosse Matrizen)
    A= B * C; // Mit Rekursion + Aufrollen in MTL4

    A+= B * D; // Inkrementiere A mit Produkt von B und D
    A-= B * D; // Dekrementiere ...
}
```

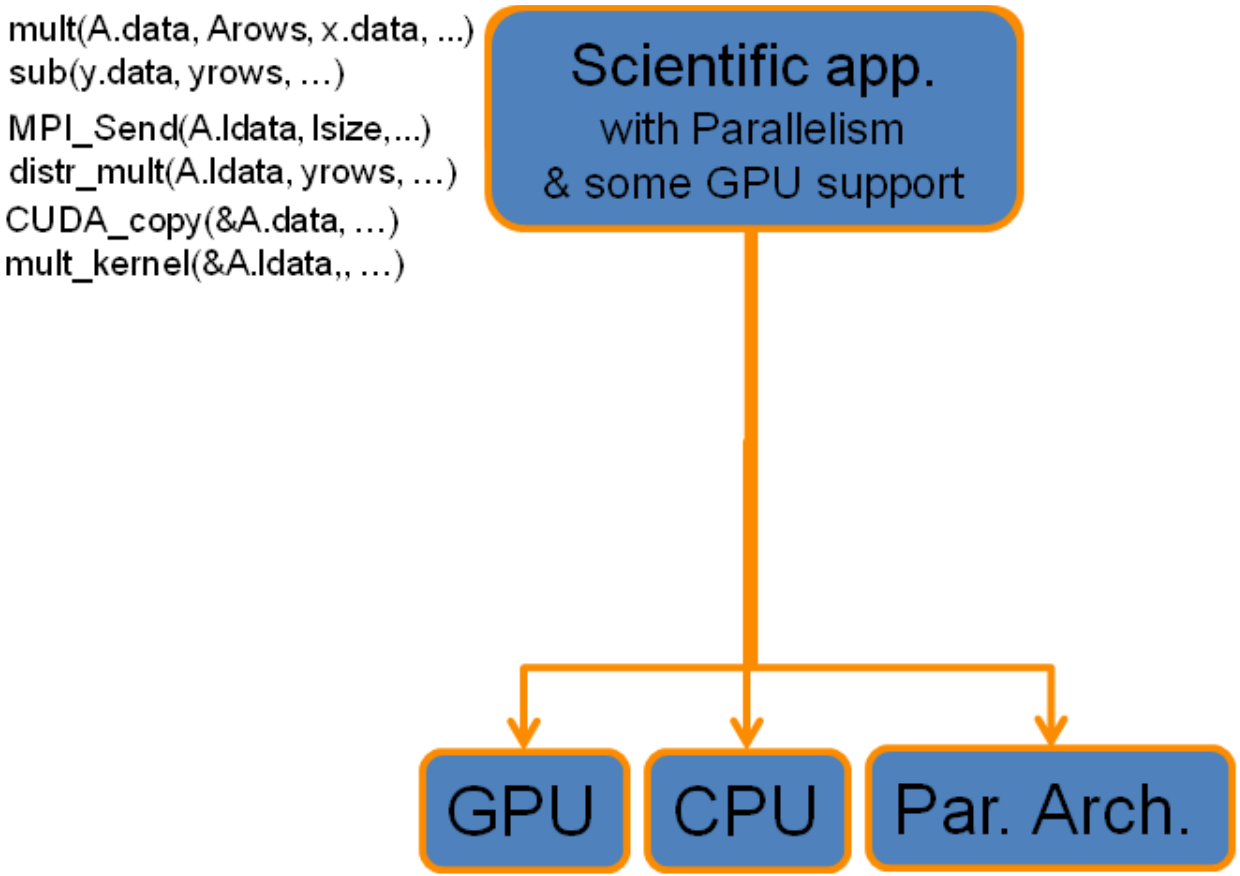
```
template <typename LinearOperator, typename SpaceX, typename SpaceB,
          typename Preconditioner, typename Iteration>
int cg(const LinearOperator& A, SpaceX& x, const SpaceB& b,
       const Preconditioner& L, Iteration& iter)
{
  typedef typename mtl::Collection<SpaceX>::value_type Scalar;
  Scalar rho(0), rho_1(0), alpha(0);
  SpaceX p(resource(x)), q(resource(x)), r(resource(x)), z(resource(x));

  r = b - A*x;
  while (! iter.finished(r)) {
      ++iter;
      z = solve(L, r);
      rho = dot(r, z);

      if (iter.first())
          p = z;
      else
          p = z + (rho / rho_1) * p;

      q = A * p;
      alpha = rho / dot(p, q);

      x += alpha * p;
      r -= alpha * q;
      rho_1 = rho;
  }
  return iter;
}
```

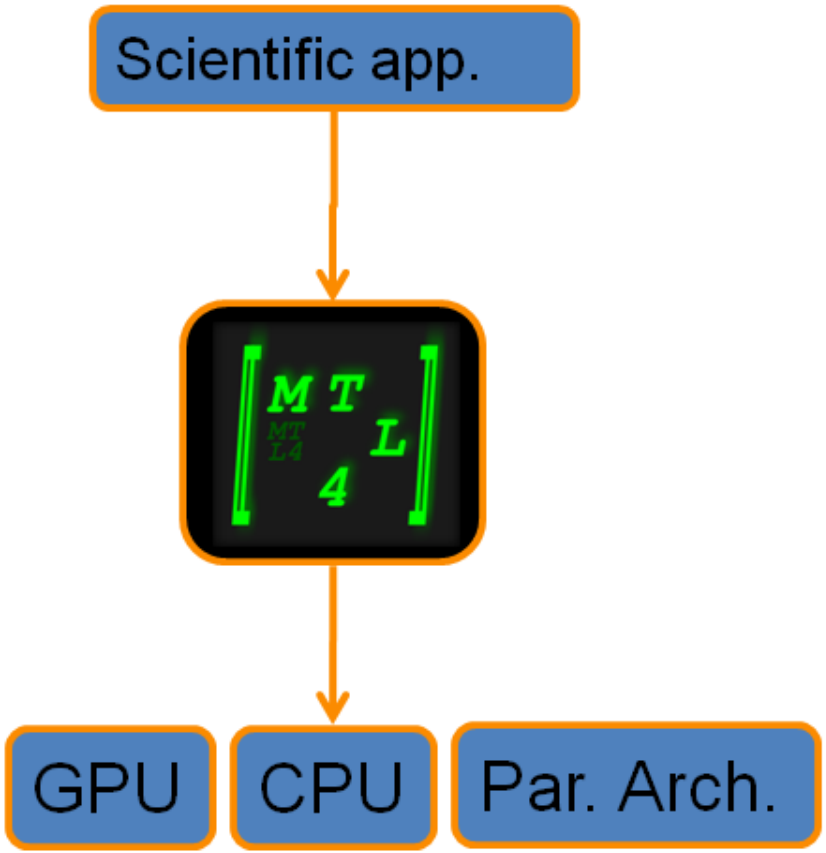
mult(A.data, Arows, x.data, ...)
sub(y.data, yrows, ...)

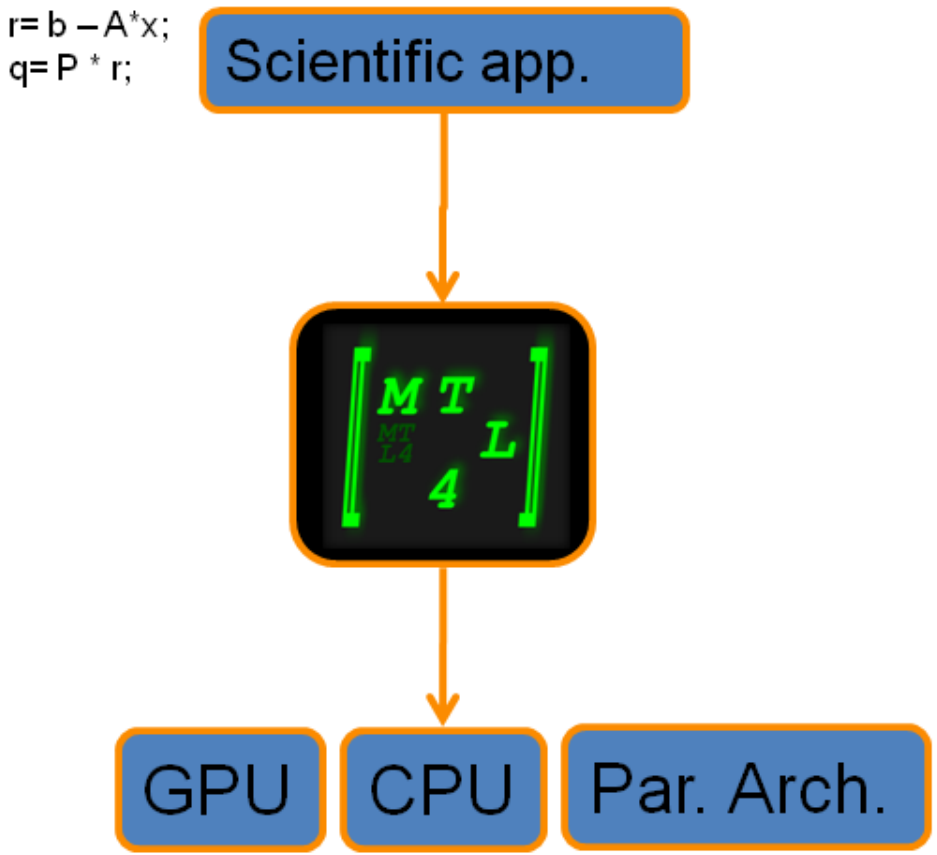


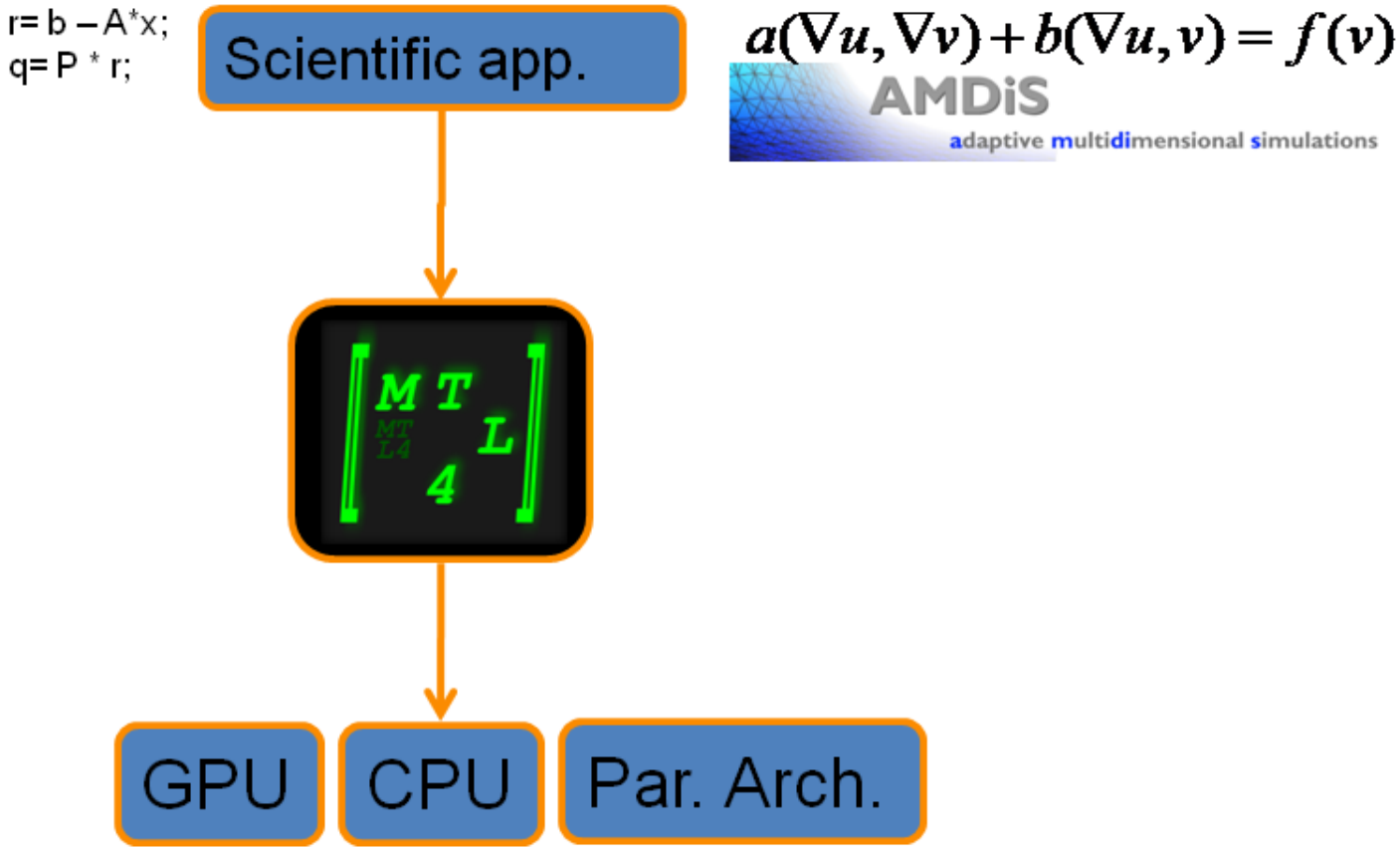
mult(A.data, Arows, x.data, ...)
sub(y.data, yrows, ...)
MPI_Send(A.ldata, lsize,...)
distr_mult(A.ldata, yrows, ...)

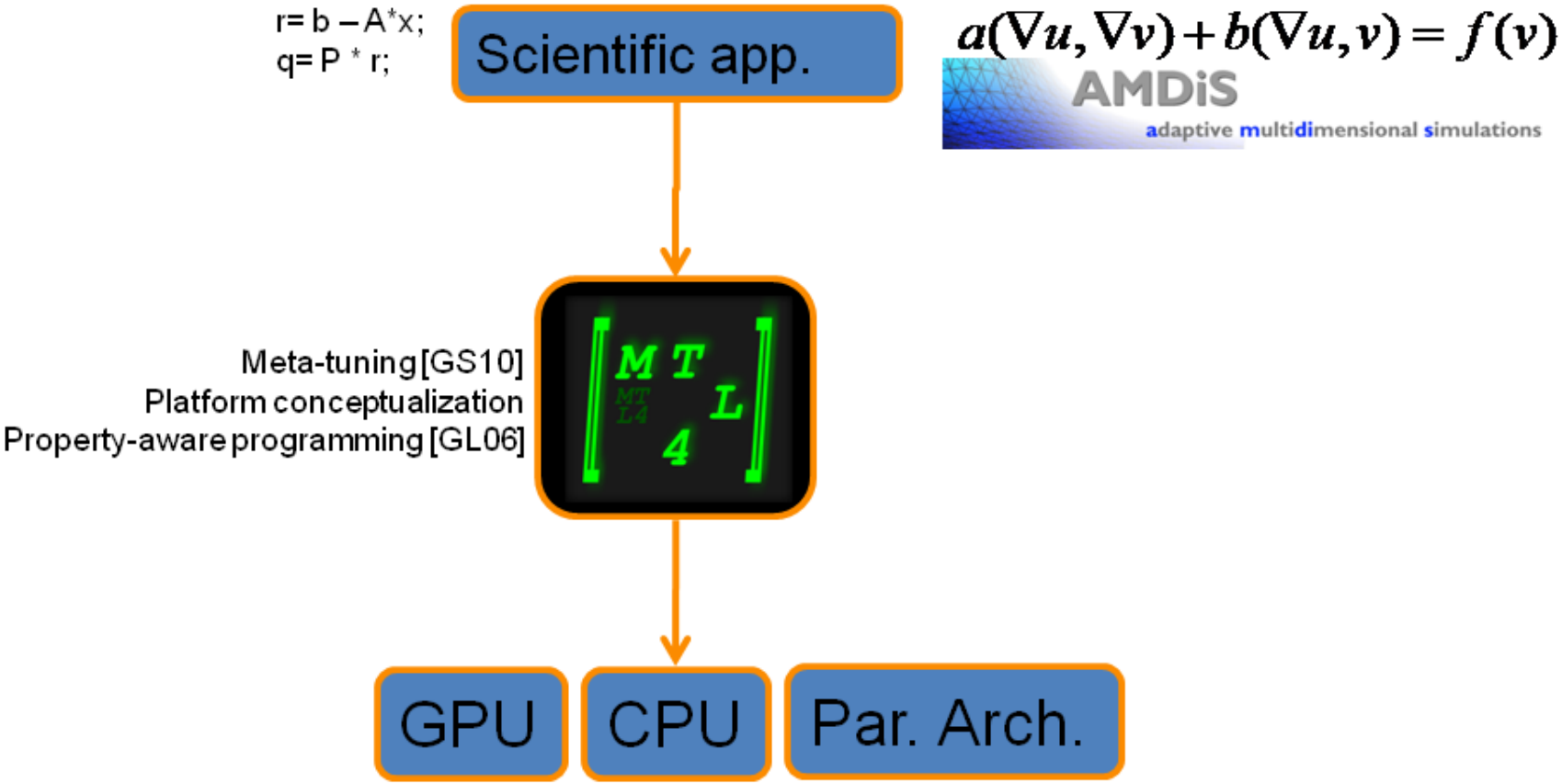


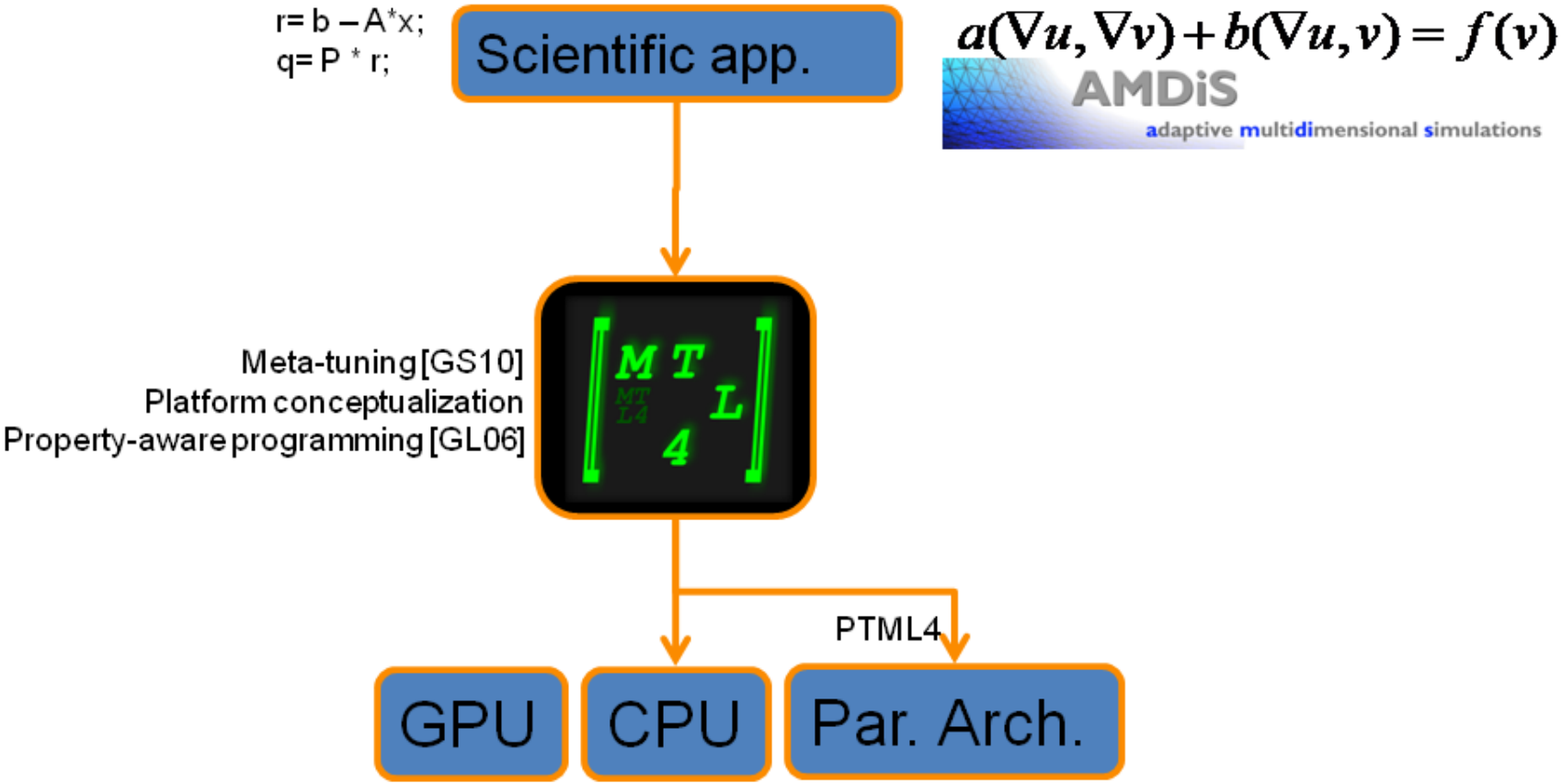

mult(A.data, Arows, x.data, ...)
sub(y.data, yrows, ...)
MPI_Send(A.ldata, lsize,...)
distr_mult(A.ldata, yrows, ...)
CUDA_copy(&A.data, ...)
mult_kernel(&A.ldata,, ...)
Scientific app.
with Parallelism
& some GPU support
GPU
CPU
Par. Arch.

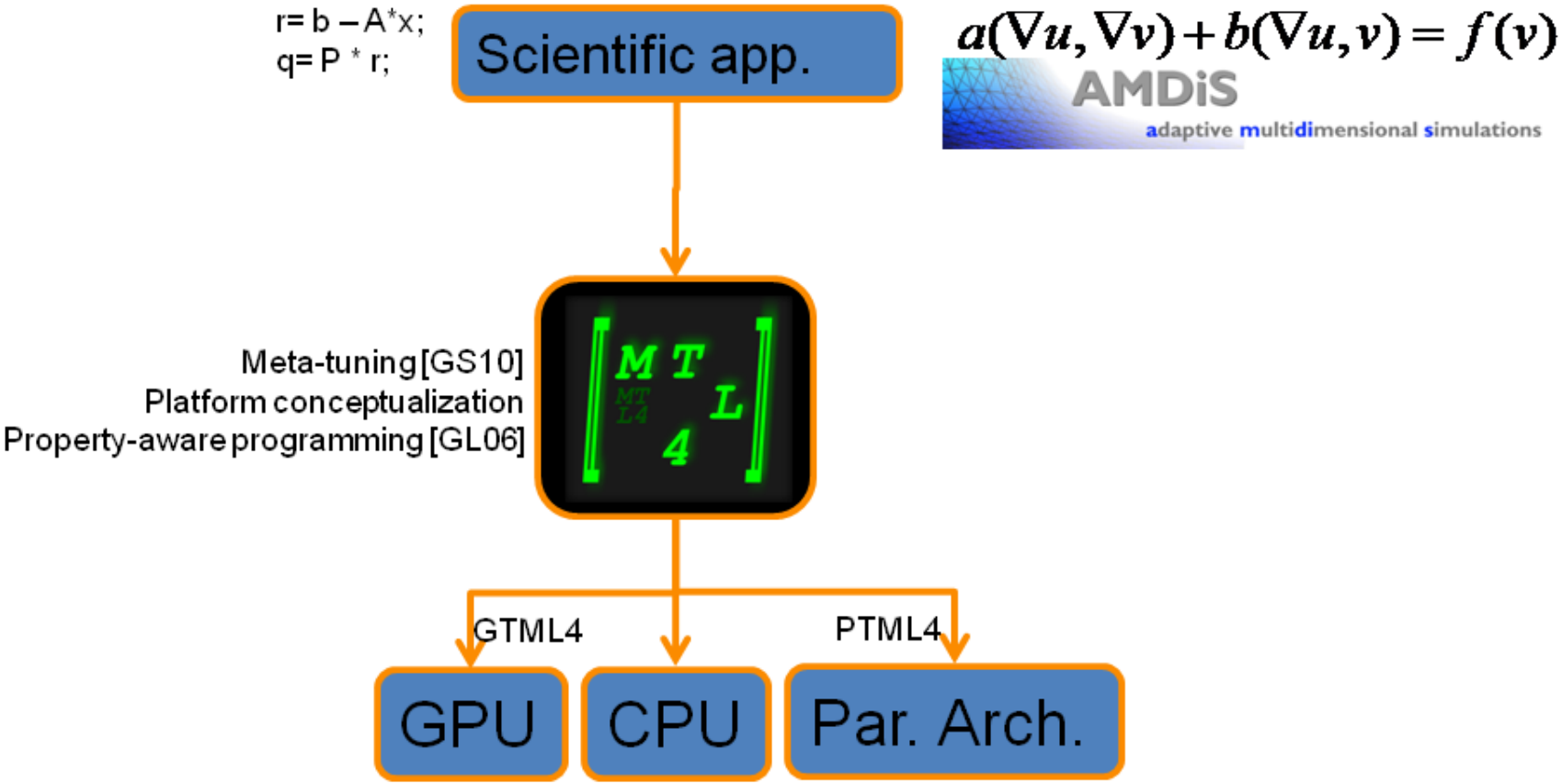

Scientific app.
GPU
CPU
Par. Arch.

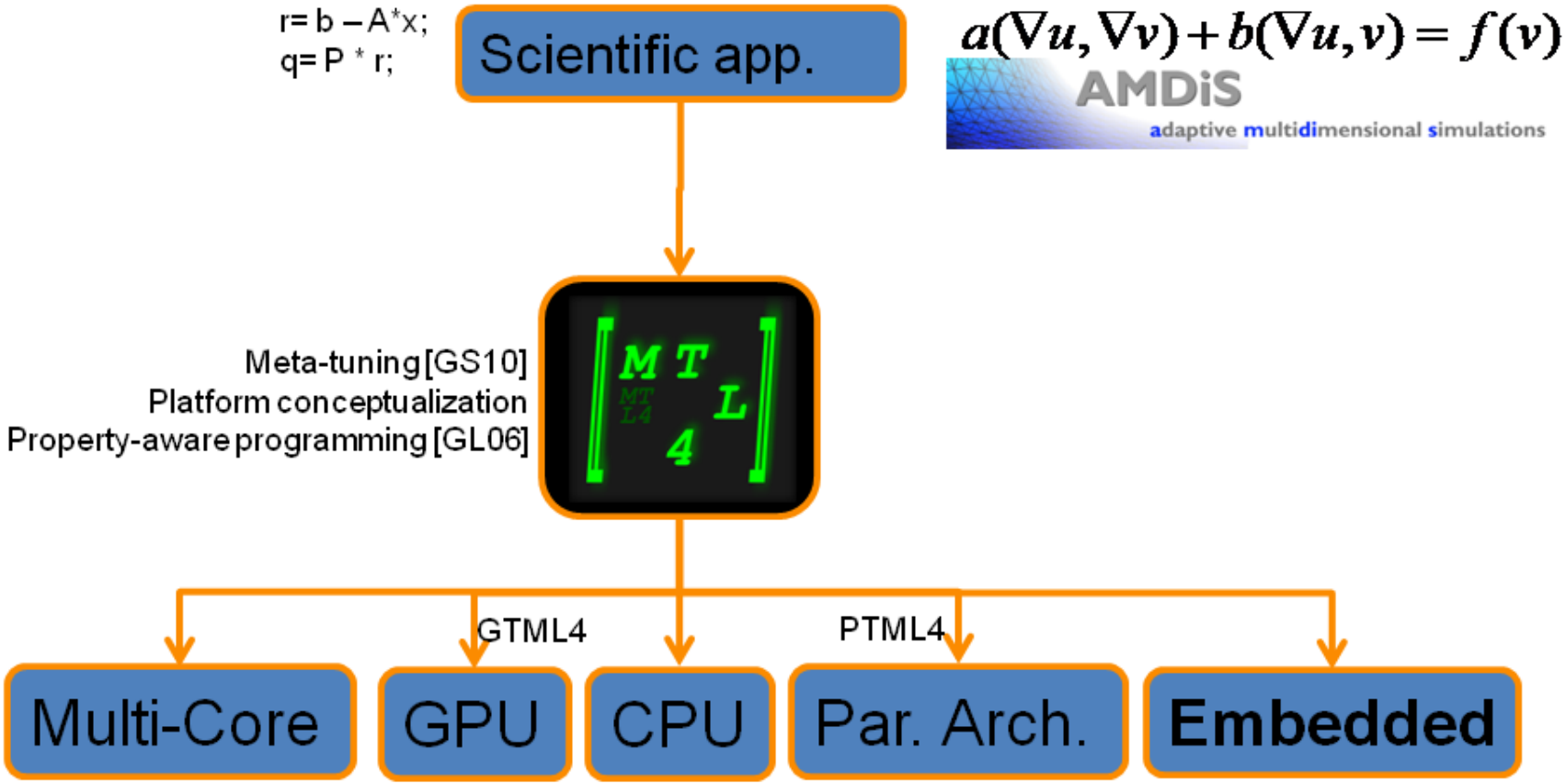

r= b – A*x;
q= P * r;
Scientific app.
GPU
CPU
Par. Arch.

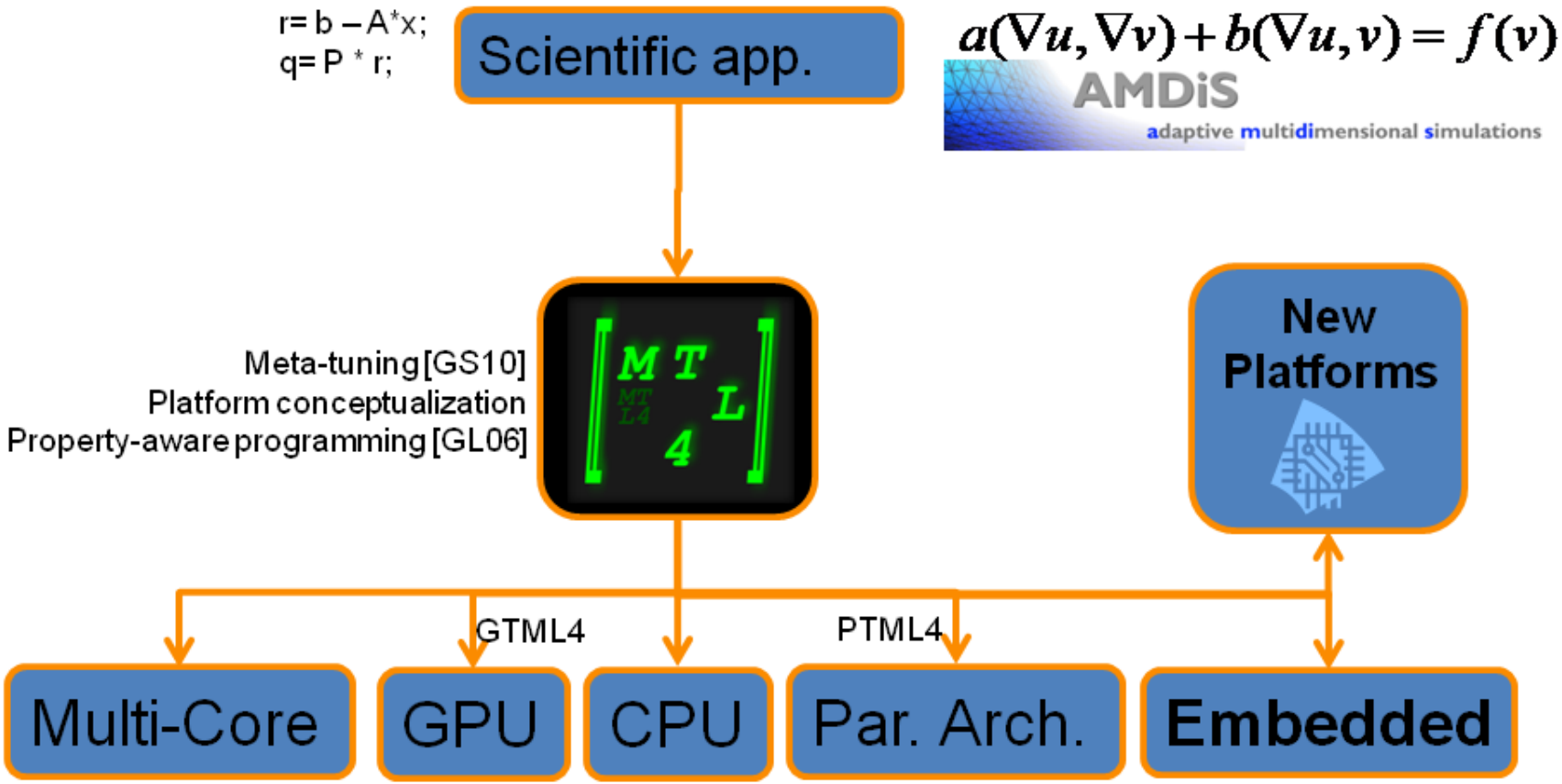

r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
MTL4
GPU
CPU
Par. Arch.


r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
Meta-tuning [GS10]
Platform conceptualization
Property-aware programming [GL06]
MTL 4
GPU
CPU
Par. Arch.


r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
Meta-tuning [GS10]
Platform conceptualization
Property-aware programming [GL06]
MTL 4
PTML4
GPU
CPU
Par. Arch.


r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
Meta-tuning [GS10]
Platform conceptualization
Property-aware programming [GL06]
MTL4
GTML4
PTML4
GPU
CPU
Par. Arch.


r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
Meta-tuning [GS10]
Platform conceptualization
Property-aware programming [GL06]
MTL4
GTML4
PTML4
Multi-Core
GPU
CPU
Par. Arch.
Embedded


r= b – A*x;
q= P * r;
Scientific app.
$a(\nabla u, \nabla v) + b(\nabla u, v) = f(v)$
AMDiS
adaptive multidimensional simulations
Meta-tuning [GS10]
Platform conceptualization
Property-aware programming [GL06]
MTL4
New Platforms
GTML4
PTML4
Multi-Core
GPU
CPU
Par. Arch.
Embedded

```
int main(int argc, char* argv[])
{
    using namespace mtl;
    par::environment env(argc, argv);

    matrix<float, sparse> A, B(12, 12), C;
    laplacian_setup(A, 3, 4);
    B= 2.0;

    C= 2.0 * A - B / 2.0;
    C+= A / 4.0 + conj(B);
    C+= A * B;

    return 0;
}
```

```
int main(int argc, char* argv[])
{
    using namespace mtl;
    using vector_type= mtl::vector<float>;
    using matrix_type= matrix<float>;

    par::environment env(argc, argv);
    vector_type b(12, 1.0);
    matrix_type A;
    laplacian_setup(A, 4, 3);

    auto migration= parmetis_migration(A);
    vector_type b2(b, migration), x2(resource(b2));
    matrix_type A2(A, migration);
    // Solve A2 * x2 = b2

    vector_type x(x2, reverse(migration));
}
```

Anwendung Kristallwachstum

- Parallel auf zwei Cloud-Installationen in China,
- Parametrisierung mühsamer als Programmierung,
- `https://www.youtube.com/watch?v=8v6snbjC38Q`
- `https://www.youtube.com/watch?v=xv9J0UCieNQ`

```
__global__ void
add(int *a, int *b, int *c ) {
  int tid = blockIdx.x; // handle the data at this index
  if (tid < N)
    c[tid] = a[tid] + b[tid];
}
#define N 10
int main(void) {
    int a[N], b[N], c[N];
    int *dev_a, *dev_b, *dev_c;
    HANDLE_ERROR( cudaMalloc( (void**)&dev_a, N * sizeof(int) ) );
    HANDLE_ERROR( cudaMalloc( (void**)&dev_b, N * sizeof(int) ) );
    HANDLE_ERROR( cudaMalloc( (void**)&dev_c, N * sizeof(int) ) );

    for (int i=0; i<N; i++) {
      a[i] = -i; b[i] = i * i;
    }
    HANDLE_ERROR( cudaMemcpy( dev_a, a, N * sizeof(int), cudaMemcpyHostToDevice ) );
    HANDLE_ERROR( cudaMemcpy( dev_b, c, N * sizeof(int), cudaMemcpyHostToDevice ) );
    HANDLE_ERROR( cudaMemcpy( dev_c, c, N * sizeof(int), cudaMemcpyHostToDevice ) );

    add<<<N,1>>>( dev_a, dev_b, dev_c );

    HANDLE_ERROR( cudaMemcpy( c, dev_c, N * sizeof(int), cudaMemcpyDeviceToHost ) );
    for(int i=0; i<N; i++)
      printf("%d + %d = %d\n", a[i], b[i], c[i] );

    cudaFree( dev_a ); cudaFree( dev_b ); cudaFree( dev_c );
    return 0;
}
```

Abstrakte (CUDA-) Programmierung

```
#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int argc, char** argv)
{
    const int N= 10;
    mtl::dense_vector<int> a(N), b(N), c;
    for (int i: irange(0, N))
        a[i]= -i, b[i]= i * i;
    c= a + b;
    std::cout << a << " + " << b << " = " << c << "\n";
    return 0;
}
```

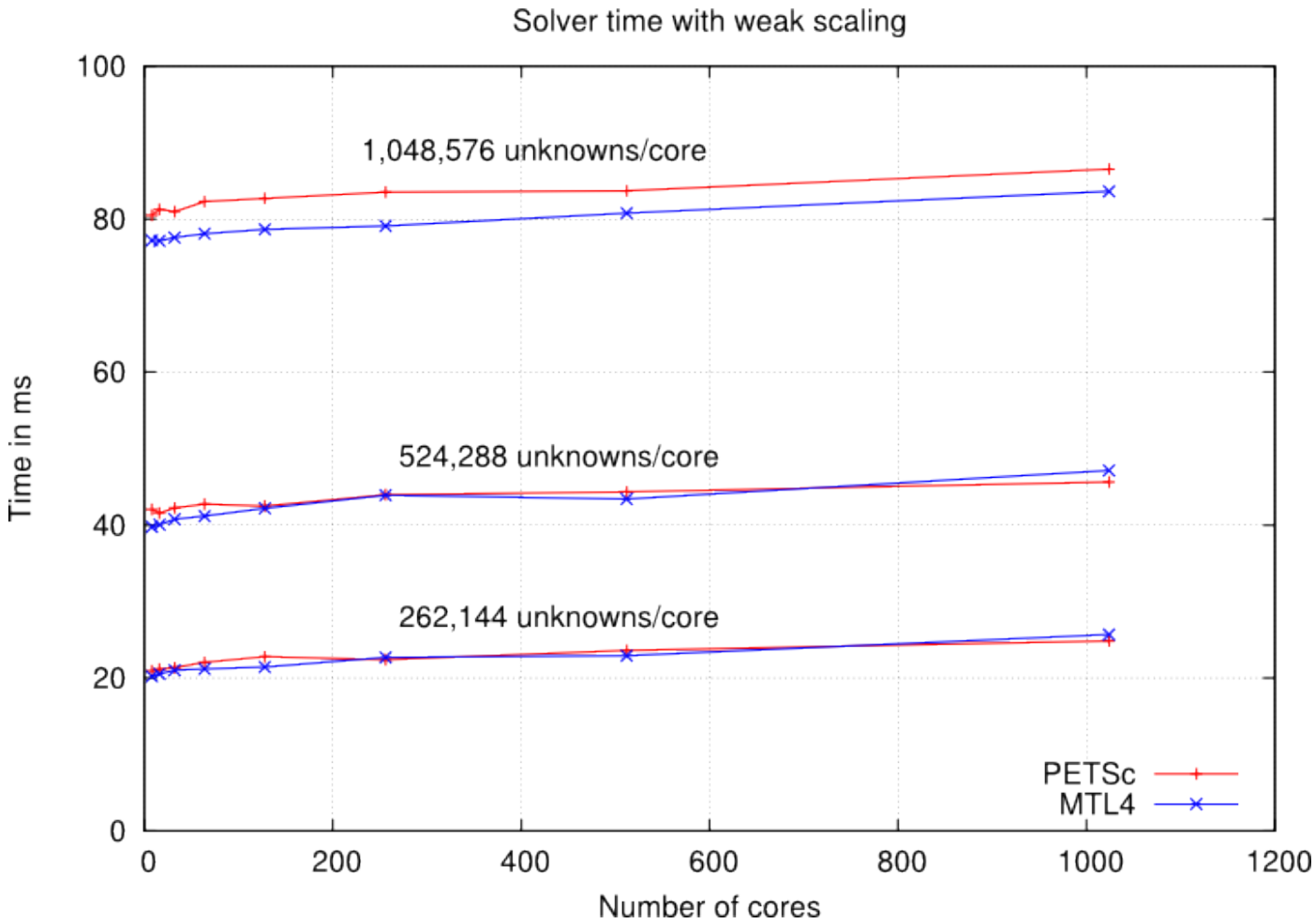


Abbildung : Vergleich parallele CG-Performance mit PETSc

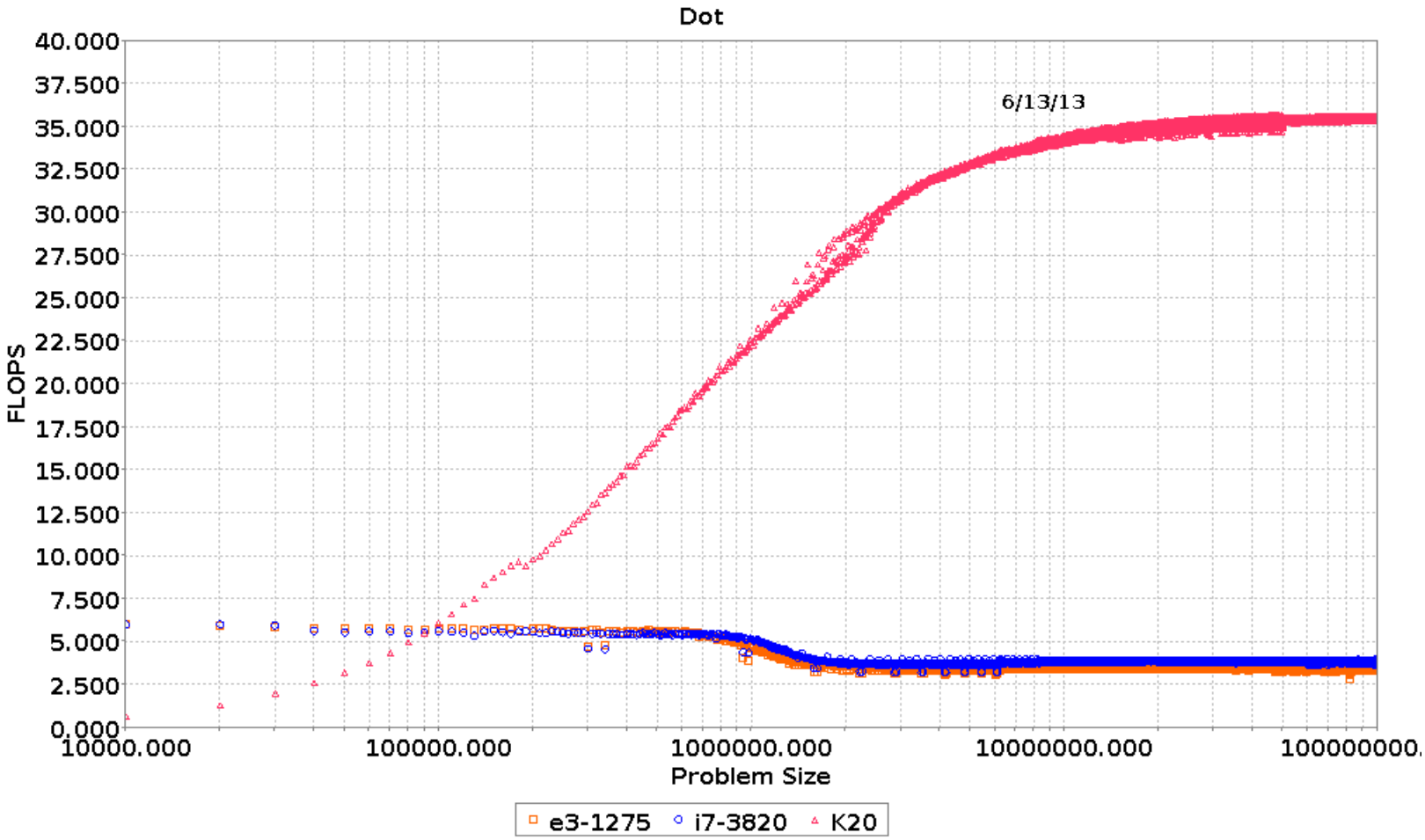


Abbildung : Skalarprodukt auf CPU und GPU

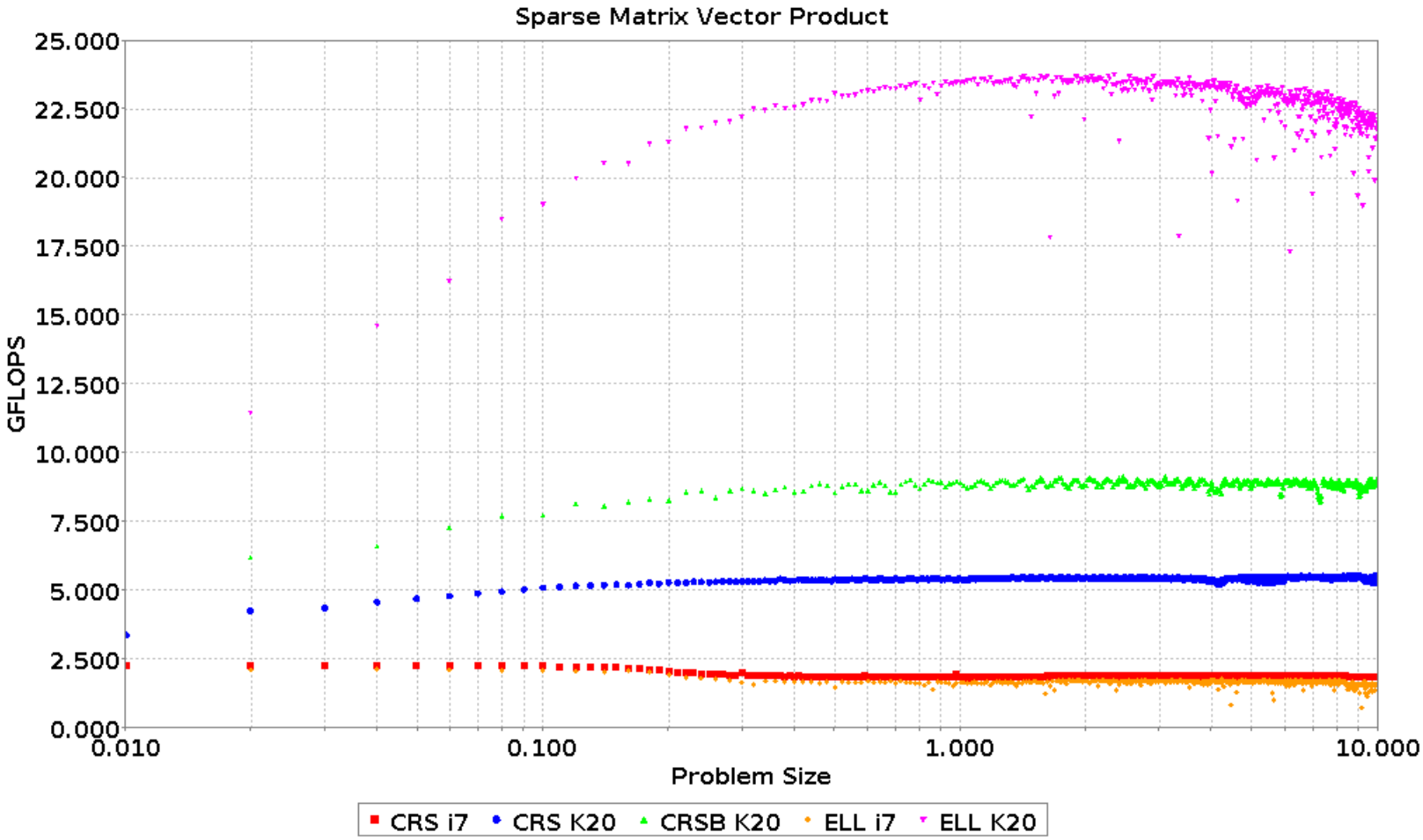


Abbildung : Matrix-Vektor-Produkt

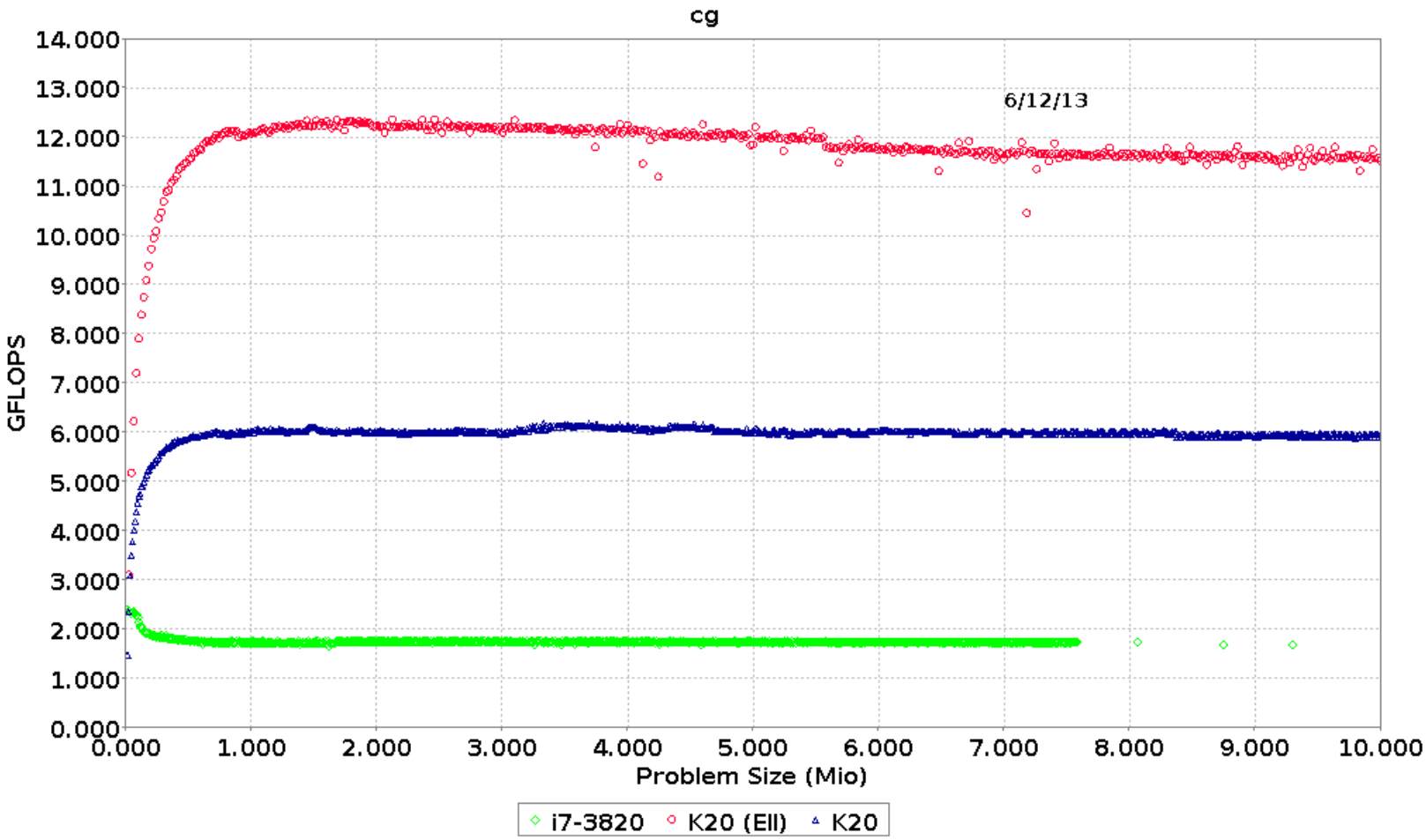
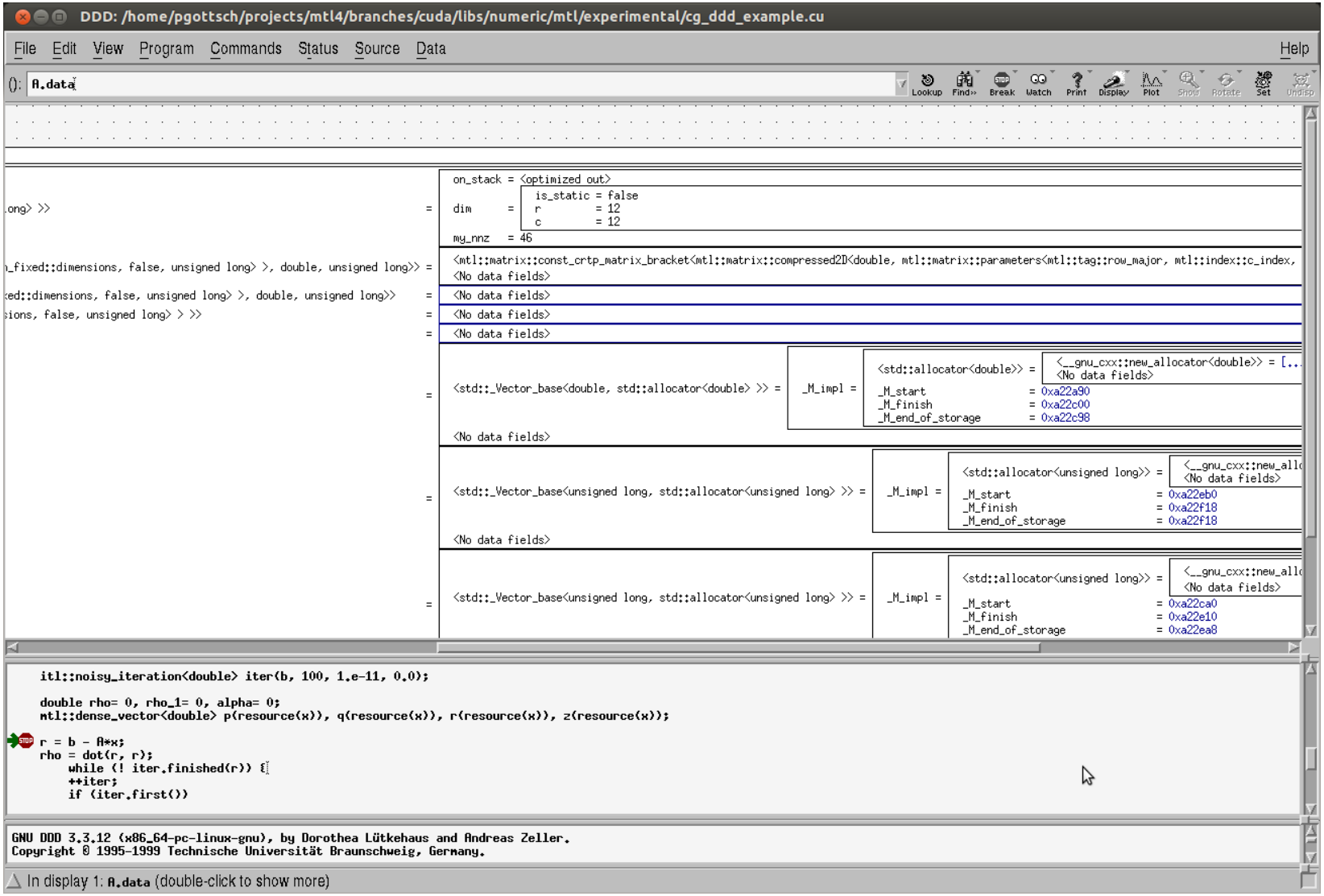
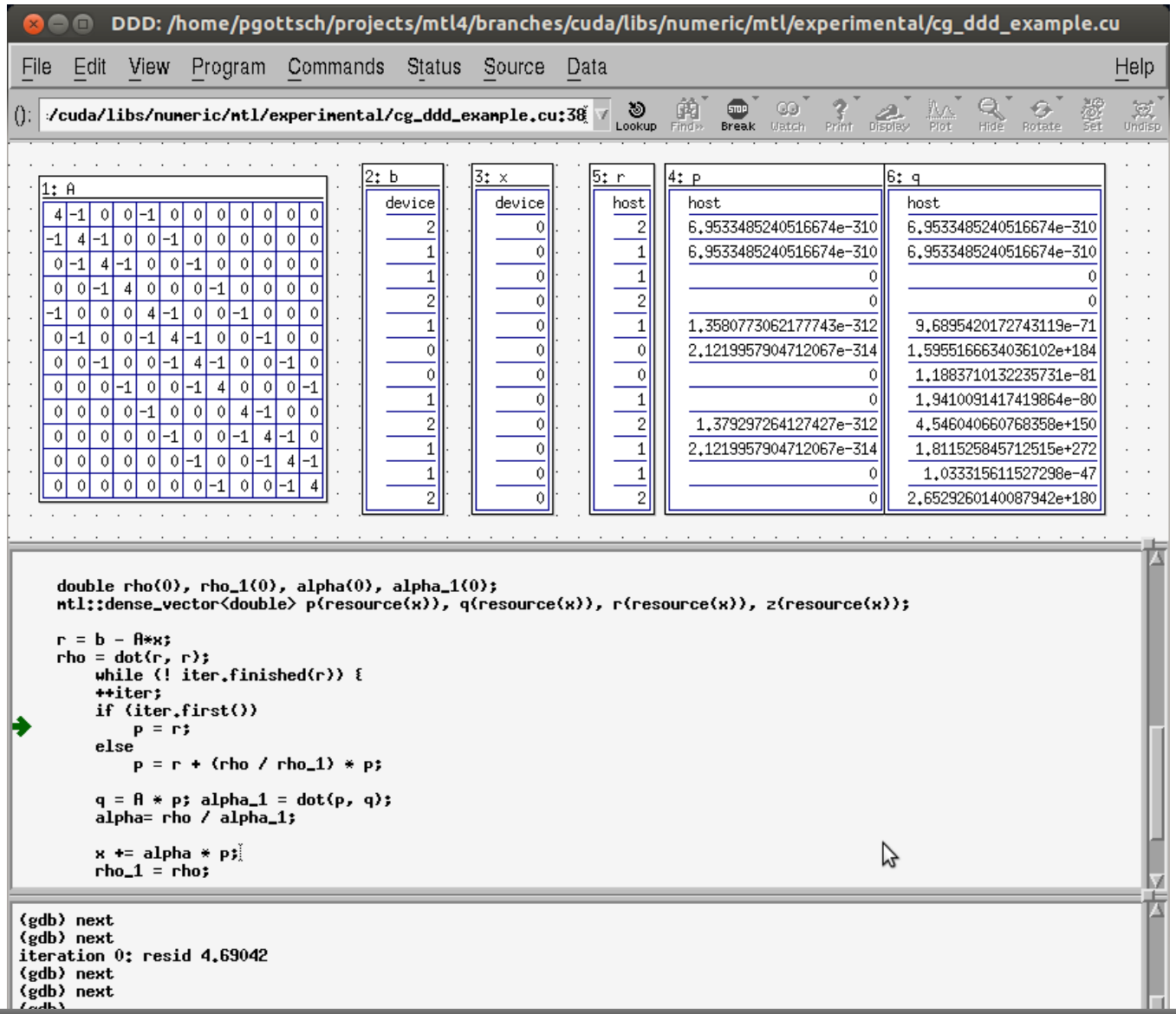
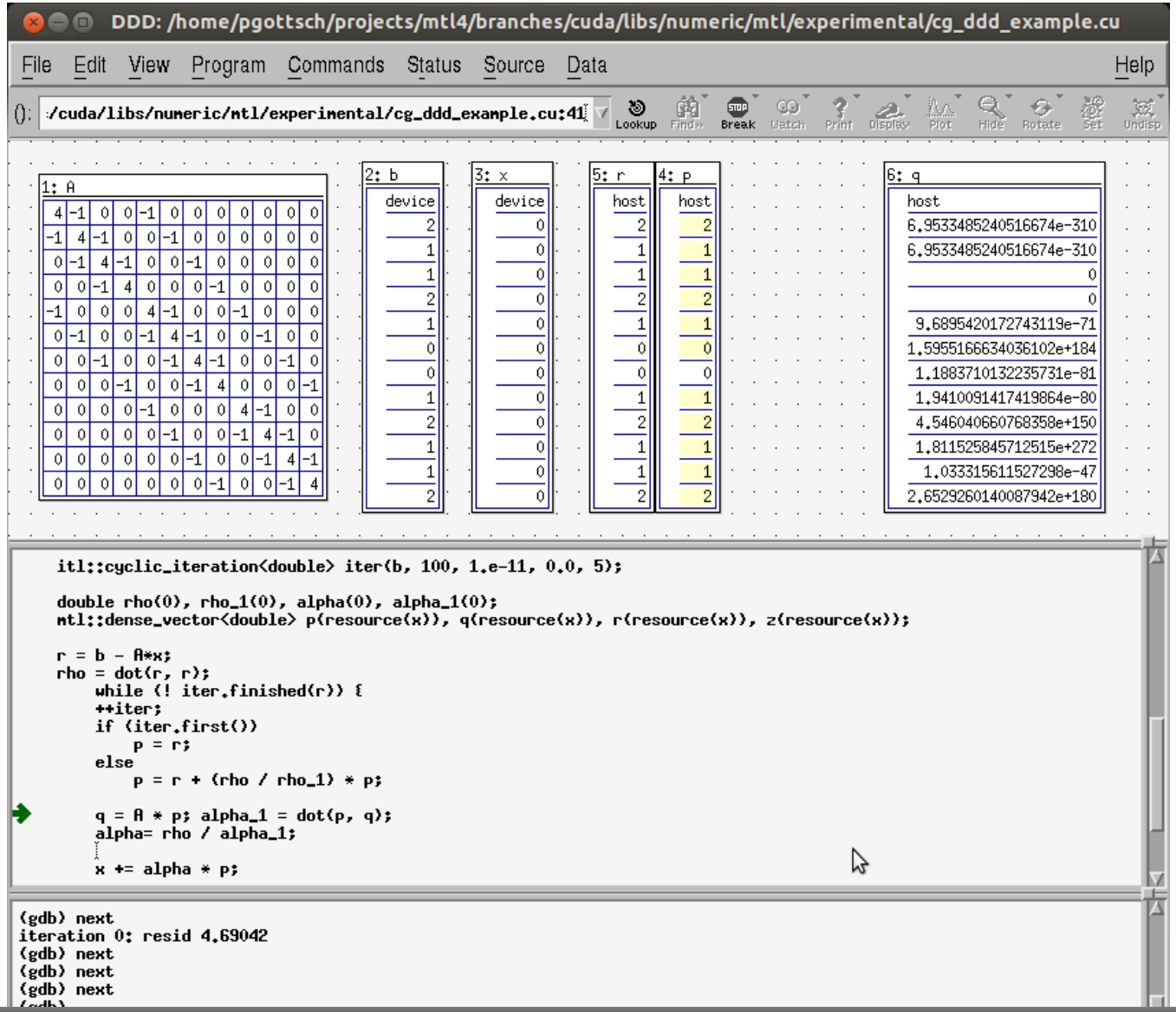
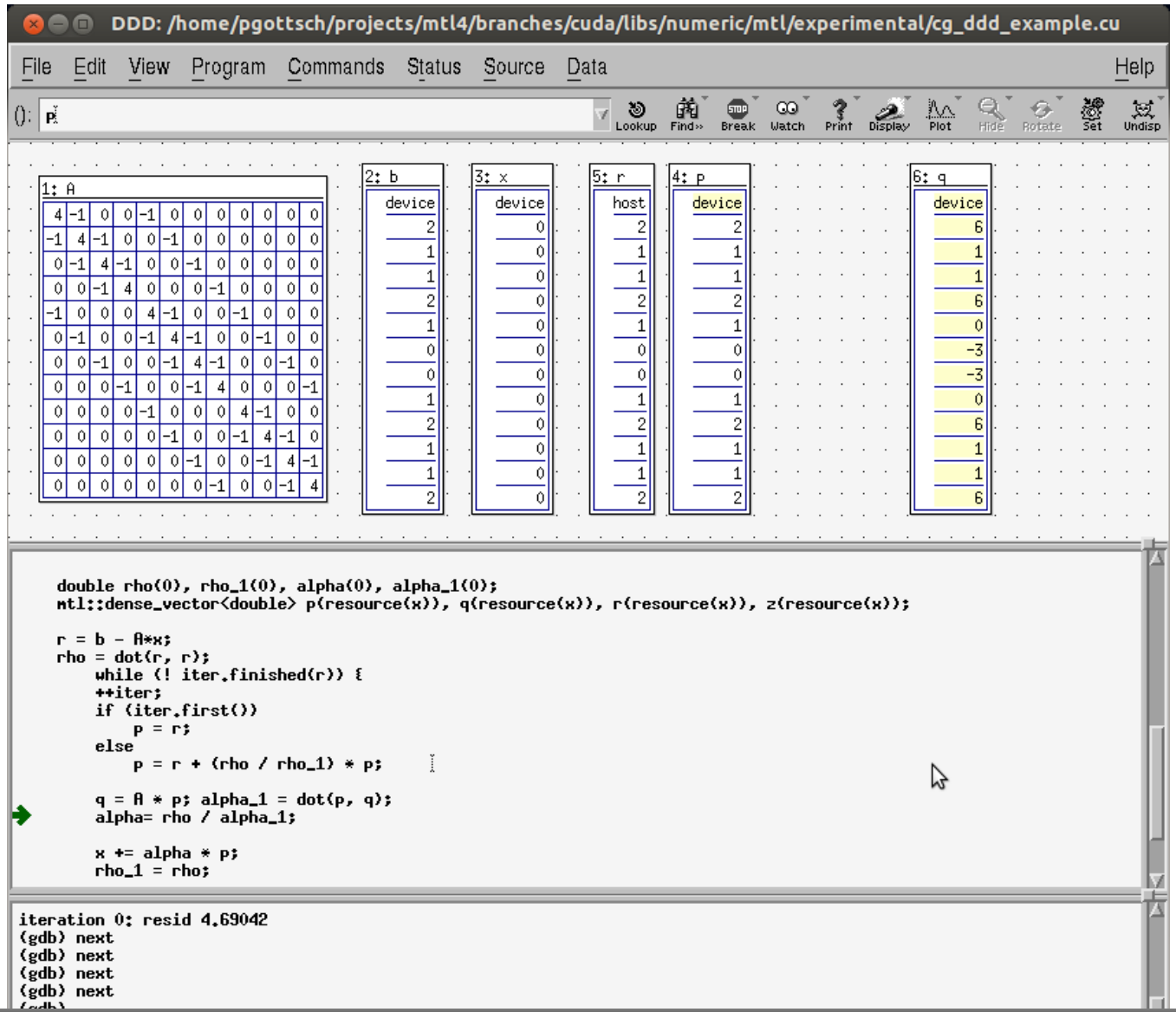


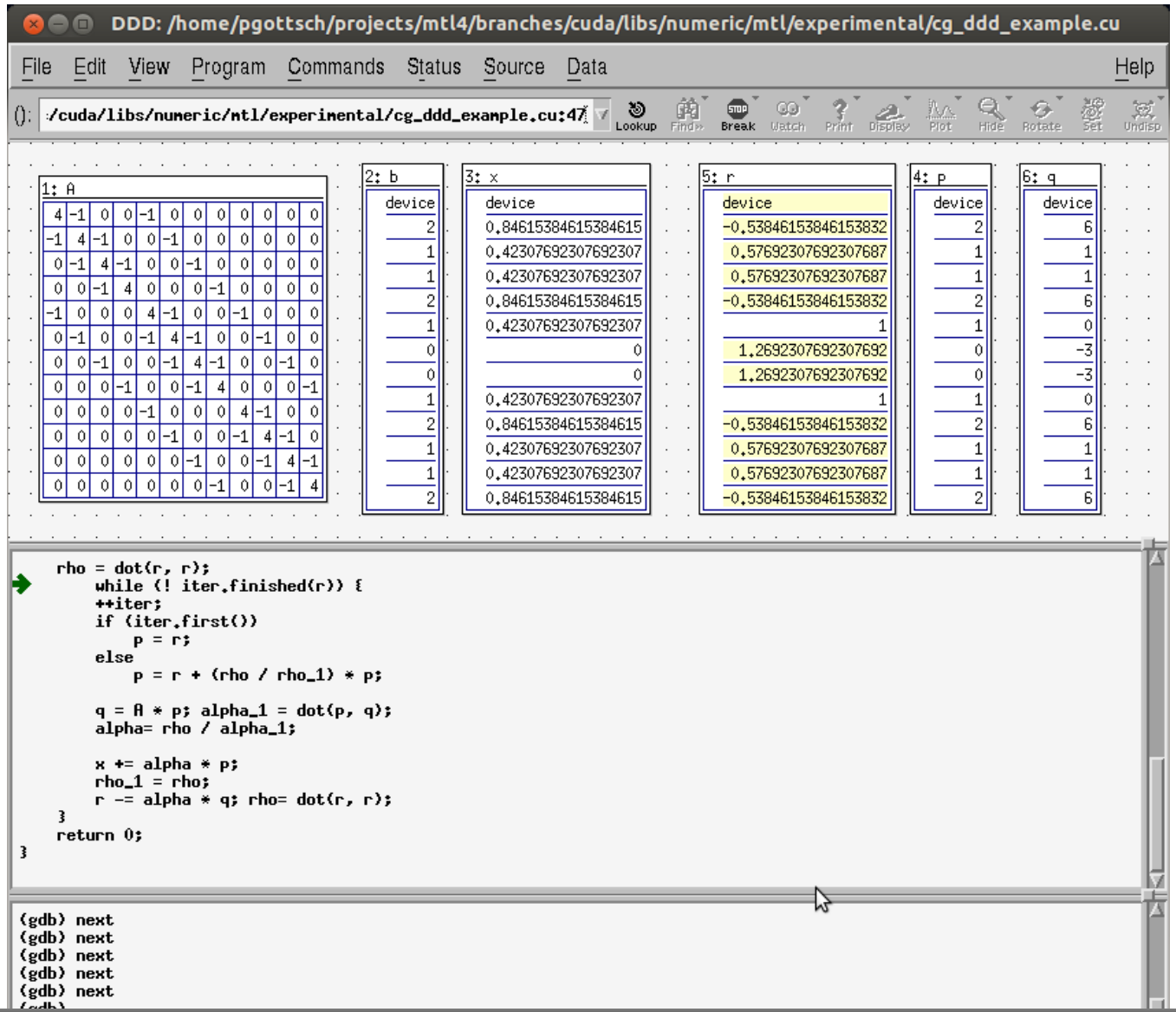
Abbildung : Konjugierte Gradienten

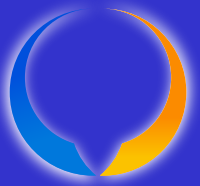

DDD: /home/pgottsch/projects/mtl4/branches/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu
File Edit View Program Commands Status Source Data Help
A.data
Lookup Find Break Watch Print Display Plot Show Rotate Set Undisp
on_stack = <optimized out>
dim = is_static = false
r = 12
c = 12
my_nnz = 46
<No data fields>
_M_impl =
_M_start = 0xa22a90
_M_finish = 0xa22c00
_M_end_of_storage = 0xa22c98
_M_start = 0xa22eb0
_M_finish = 0xa22f18
_M_end_of_storage = 0xa22f18
_M_start = 0xa22ca0
_M_finish = 0xa22e10
_M_end_of_storage = 0xa22ea8
itl::noisy_iteration<double> iter(b, 100, 1.e-11, 0.0);
double rho= 0, rho_1= 0, alpha= 0;
mtl::dense_vector<double> p(resource(x)), q(resource(x)), r(resource(x)), z(resource(x));
r = b - A*x;
rho = dot(r, r);
while (! iter.finished(r)) {
++iter;
if (iter.first())
GNU DDD 3.3.12 (x86_64-pc-linux-gnu), by Dorothea Lütkehaus and Andreas Zeller.
Copyright © 1995-1999 Technische Universität Braunschweig, Germany.
In display 1: A.data (double-click to show more)


DDD: /home/pgottsch/projects/mtl4/branches/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu
File Edit View Program Commands Status Source Data Help
():
/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu:38
Lookup Find... Break Watch Print Display Plot Hide Rotate Set Undisp
1: A
4 -1 0 0 -1 0 0 0 0 0 0 0
-1 4 -1 0 0 -1 0 0 0 0 0 0
0 -1 4 -1 0 0 -1 0 0 0 0 0
0 0 -1 4 0 0 0 -1 0 0 0 0
-1 0 0 0 4 -1 0 0 -1 0 0 0
0 -1 0 0 -1 4 -1 0 0 -1 0 0
0 0 -1 0 0 -1 4 -1 0 0 -1 0
0 0 0 -1 0 0 -1 4 0 0 0 -1
0 0 0 0 -1 0 0 0 4 -1 0 0
0 0 0 0 0 -1 0 0 -1 4 -1 0
0 0 0 0 0 0 -1 0 0 -1 4 -1
0 0 0 0 0 0 0 -1 0 0 -1 4
2: b
device
2 1 1 2 1 0 0 1 2 1 1 2
3: x
device
0 0 0 0 0 0 0 0 0 0 0 0
5: r
host
2 1 1 2 1 0 0 1 2 1 1 2
4: p
host
6.953485240516674e-310
6.953485240516674e-310
0
0
1.358077306217743e-312
2.121995790471206 7e-314
0
0
1.379297264127427e-312
2.121995790471206 7e-314
0
0
6: q
host
6.953485240516674e-310
6.953485240516674e-310
0
0
9.689542017274311 9e-71
1.595516663403610 2e+184
1.188371013223573 1e-81
1.941009141741986 4e-80
4.546040660768358e+150
1.811525845712515e+272
1.033315611527298e-47
2.652926014008794 2e+180
double rho(0), rho_1(0), alpha(0), alpha_1(0);
mtl::dense_vector<double> p(resource(x)), q(resource(x)), r(resource(x)), z(resource(x));
r = b - A*x;
rho = dot(r, r);
while (! iter.finished(r)) {
++iter;
if (iter.first())
p = r;
else
p = r + (rho / rho_1) * p;
q = A * p; alpha_1 = dot(p, q);
alpha= rho / alpha_1;
x += alpha * p;
rho_1 = rho;
(gdb) next
(gdb) next
iteration 0: resid 4.69042
(gdb) next
(gdb) next


DDD: /home/pgottsch/projects/mtl4/branches/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu
File Edit View Program Commands Status Source Data Help
/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu:41
Lookup Find Break Watch Print Display Plot Hide Rotate Set Undisp
1: A
2: b
device
3: x
device
5: r
host
4: p
host
6: q
host
6.9533485240516674e-310
6.9533485240516674e-310
9.6895420172743119e-71
1.5955166634036102e+184
1.1883710132235731e-81
1.9410091417419864e-80
4.546040660768358e+150
1.811525845712515e+272
1.033315611527298e-47
2.6529260140087942e+180
itl::cyclic_iteration<double> iter(b, 100, 1.e-11, 0.0, 5);
double rho(0), rho_1(0), alpha(0), alpha_1(0);
mtl::dense_vector<double> p(resource(x)), q(resource(x)), r(resource(x)), z(resource(x));
r = b - A*x;
rho = dot(r, r);
while (! iter.finished(r)) {
++iter;
if (iter.first())
p = r;
else
p = r + (rho / rho_1) * p;
q = A * p; alpha_1 = dot(p, q);
alpha= rho / alpha_1;
x += alpha * p;
(gdb) next
iteration 0: resid 4.69042
(gdb) next
(gdb) next
(gdb) next


DDD: /home/pgottsch/projects/mtl4/branches/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu
File Edit View Program Commands Status Source Data Help
Lookup Find» Break Watch Print Display Plot Hide Rotate Set Undisp
1: A
2: b
3: x
5: r
4: p
6: q
device
host
double rho(0), rho_1(0), alpha(0), alpha_1(0);
mtl::dense_vector<double> p(resource(x)), q(resource(x)), r(resource(x)), z(resource(x));
r = b - A*x;
rho = dot(r, r);
while (! iter.finished(r)) {
++iter;
if (iter.first())
p = r;
else
p = r + (rho / rho_1) * p;
q = A * p; alpha_1 = dot(p, q);
alpha= rho / alpha_1;
x += alpha * p;
rho_1 = rho;
iteration 0: resid 4.69042
(gdb) next
(gdb) next
(gdb) next
(gdb) next


DDD: /home/pgottsch/projects/mtl4/branches/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu
File Edit View Program Commands Status Source Data Help
/cuda/libs/numeric/mtl/experimental/cg_ddd_example.cu:47
Lookup Find... Break Watch Print Display Plot Hide Rotate Set Undisp
1: A
2: b
3: x
5: r
4: p
6: q
device
rho = dot(r, r);
while (! iter.finished(r)) {
++iter;
if (iter.first())
p = r;
else
p = r + (rho / rho_1) * p;
q = A * p; alpha_1 = dot(p, q);
alpha= rho / alpha_1;
x += alpha * p;
rho_1 = rho;
r -= alpha * q; rho= dot(r, r);
}
return 0;
}
(gdb) next
(gdb) next
(gdb) next
(gdb) next
(gdb) next



- FEniCS: freie Software zum automatischen Lösen differentieller Gleichungen.
  - University of Chicago, Argonne National Laboratory, Delft University of Technology, Royal Institute of Technology KTH, Simula Research Laboratory, Texas Tech University und University of Cambridge.
- Bombardier: Entwicklung neuer Flugzeuge.
- Institut Français du Pétrole: Mehrphasensimulation unterirdischer CO_2-Speicherung mittels finiter Elemente.
- Serbisches Institut für Grundwasserressourcen: dichte-getriebene Grundwasserströmungen mit stark diskontinuierlicher und anisotroper Permeabilität.