Eigen

a c++ linear algebra library

[http://eigen.tuxfamily.org]

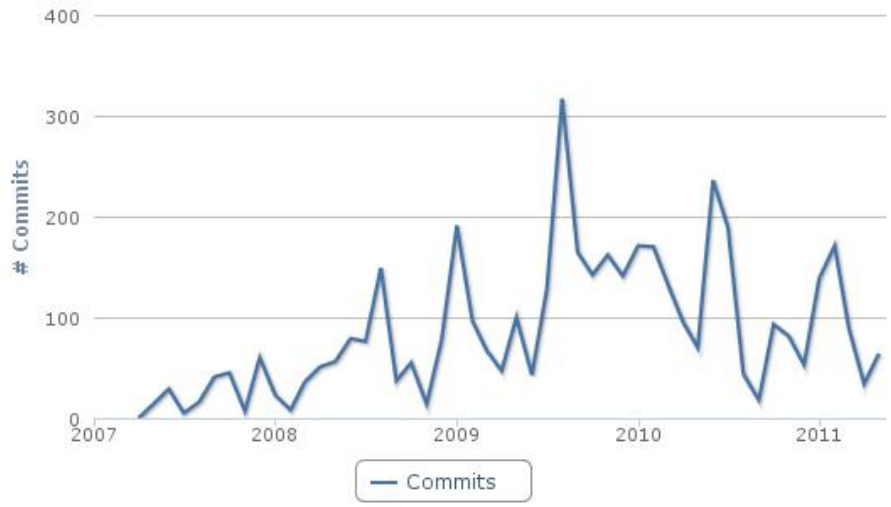
Gaël Guennebaud

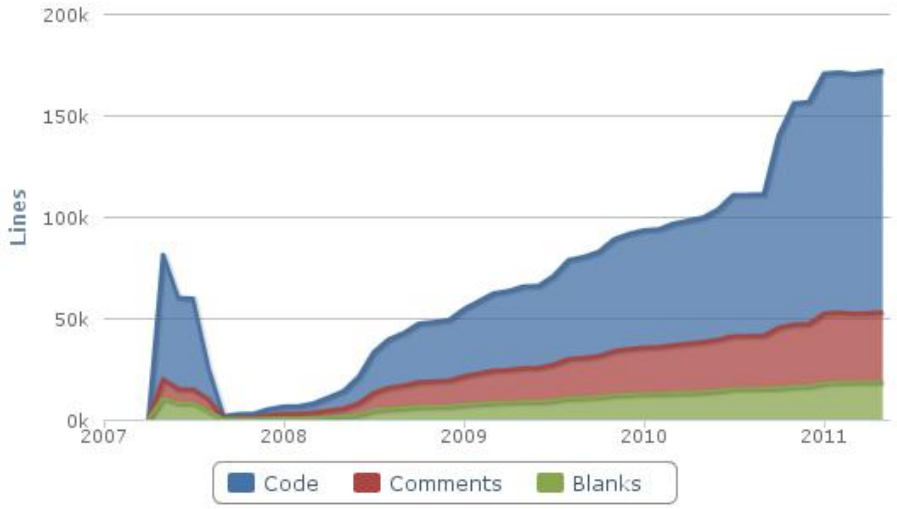
co-developed with Benoit Jacob (Mozilla)
and a bunch of handful occasional contributors (PhDs)

Inria

History

- Janv 2008: start of Eigen2
- Feb 2009: release Eigen 2.0
- Feb 2009: 1st annual meeting (3 days, ~10 people)
- March 2011: release Eigen 3.0
- March 2011: 2nd annual meeting (3 days)





Facts

- Active project with many users
  - Website: ~9000 unique visitors / month
  - Mailing list/Forum:
    - ~250 members, ~400 messages/month
- Pure C++ template library
  - header only, no binary to compile, install...
- Packaged by all Linux distributions
- Opensource: LGPL3+

→ **easy to install & distribute**

Large feature set

- Modules:
  - Core
    - Matrix and array manipulation (MatLab, 1D & 2D only)
    - Basic linear algebra (~BLAS)
  - LU, Cholesky, QR, SVD, Eigenvalues
    - Matrix decompositions and linear solvers (~Lapack)
  - Geometry (transformations, …)
  - WIP modules:
    - Sparse matrices, Automatic differentiation, Non-linear optimization, FFT, etc.

→ **“unified API” - “all-in-one”**

Optimized for both small and large objects

- Small objects
  - means fixed sizes:

```
Eigen::Matrix<float,4,4>
```

  - Malloc-free
  - Meta unrolling

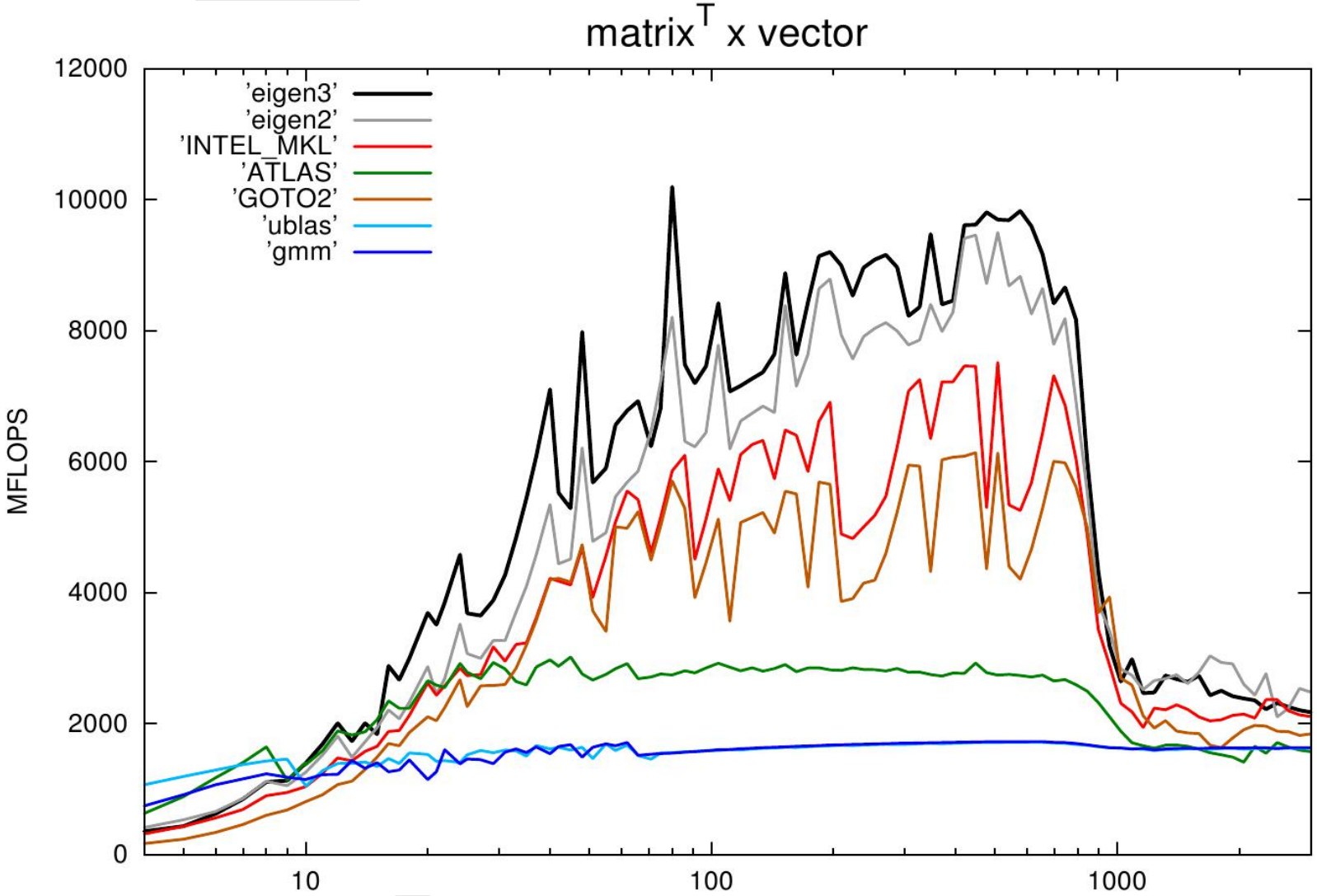
- Large objects
  - means dynamic sizes

```
Eigen::Matrix<float,Dynamic,1>
```

  - Cache friendly kernels
  - Parallelization *(OpenMP)*

- Vectorization (SIMD)
- Unified API

Performance



Multi-platforms

- Supported compilers:
  - GCC *(from 3.4 to 4.6)*, MSVC *(2005,2008,2010)*, Intel ICC, Clang/LLVM *(2.8)*
- Supported systems:
  - x86/x86_64 (Linux,Windows)
  - ARM (Linux), PowerPC
- Supported SIMD vectorization engines:
  - SSE2, SSE3, SSSE3, SSE4
  - NEON (ARM)
  - Altivec (PowerPC)

Custom scalar types

- Can use custom types everywhere
  - Exact arithmetic (rational numbers)
  - Multi-precision numbers (e.g., via mpreal++)
  - Auto-diff scalar types
  - Interval
  - Symbolic

- Example:

```
typedef Matrix<mpreal,Dynamic,Dynamic> MatrixX;
MatrixX A, B, X;
// init A and B
// solve for A.X=B using LU decomposition
X = A.lu().solve(B);
```

Summary

- Many unique features
- Goal: → ideal compromise between:
  - versatility
  - ease of use
  - performance
- in-between MatLab ↔ specialized numerical packages
- Main targets:
  - researchers, end user applications, embedded applications, education, etc.

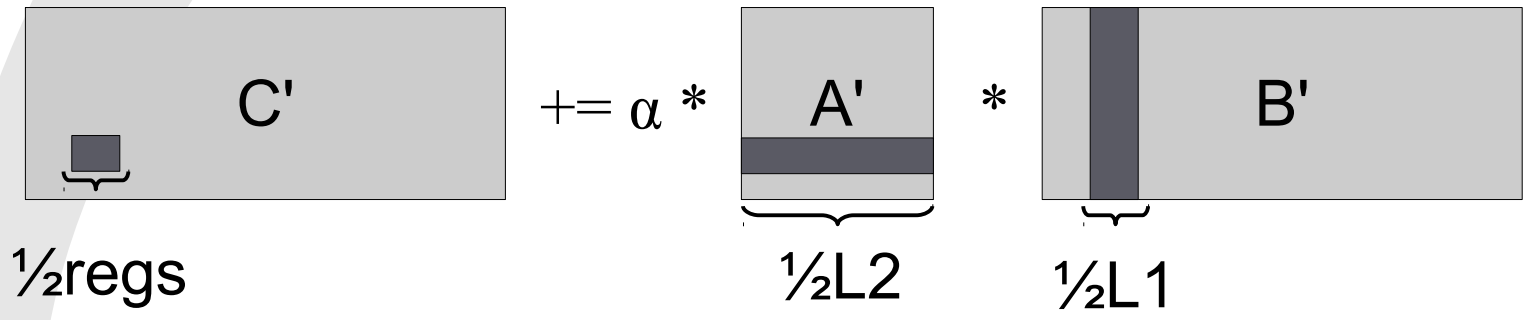
Eigen vs BLAS/Lapack

- Pros:
  - C++ friendly API
  - Matrix manipulation
  - Easy to use, install, distribute, etc.
  - Custom scalar types
  - Static allocation
  - Temporary removal
  - Auto vectorization, ARM NEON
  - Higher perf for small objects
  - etc.
- Cons:
  - Covers only most common features of lapack
  - Not fully multi-threaded yet

Inside Eigen

Matrix products
[Based on Goto's paper]

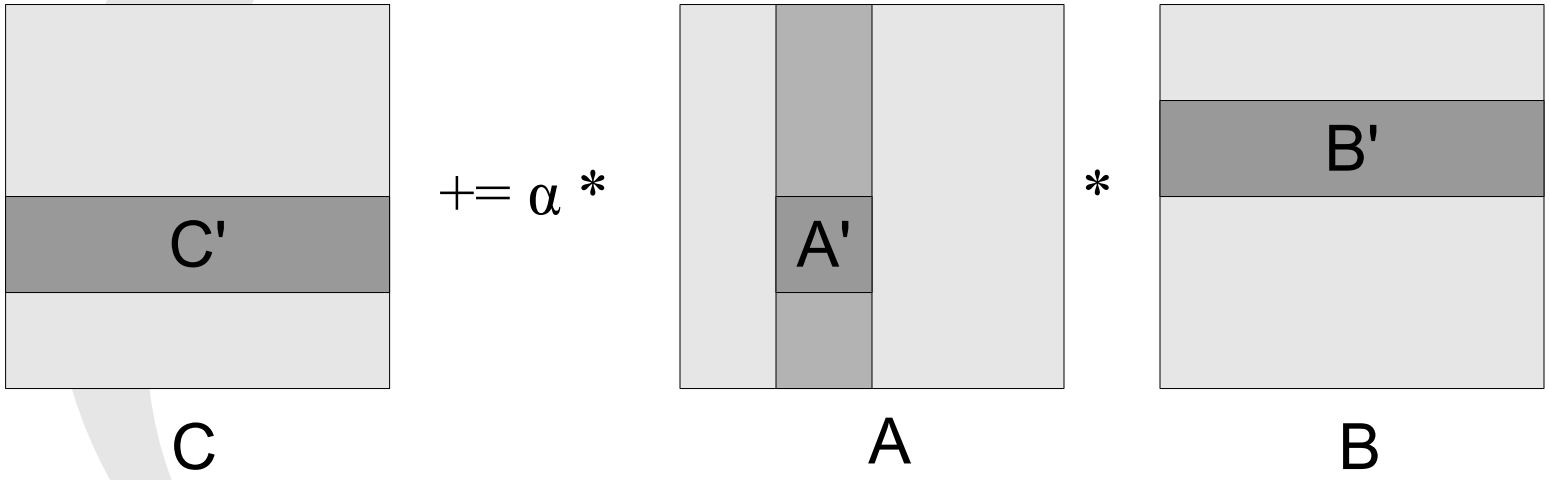
- Highly optimized Block*Panel kernel:



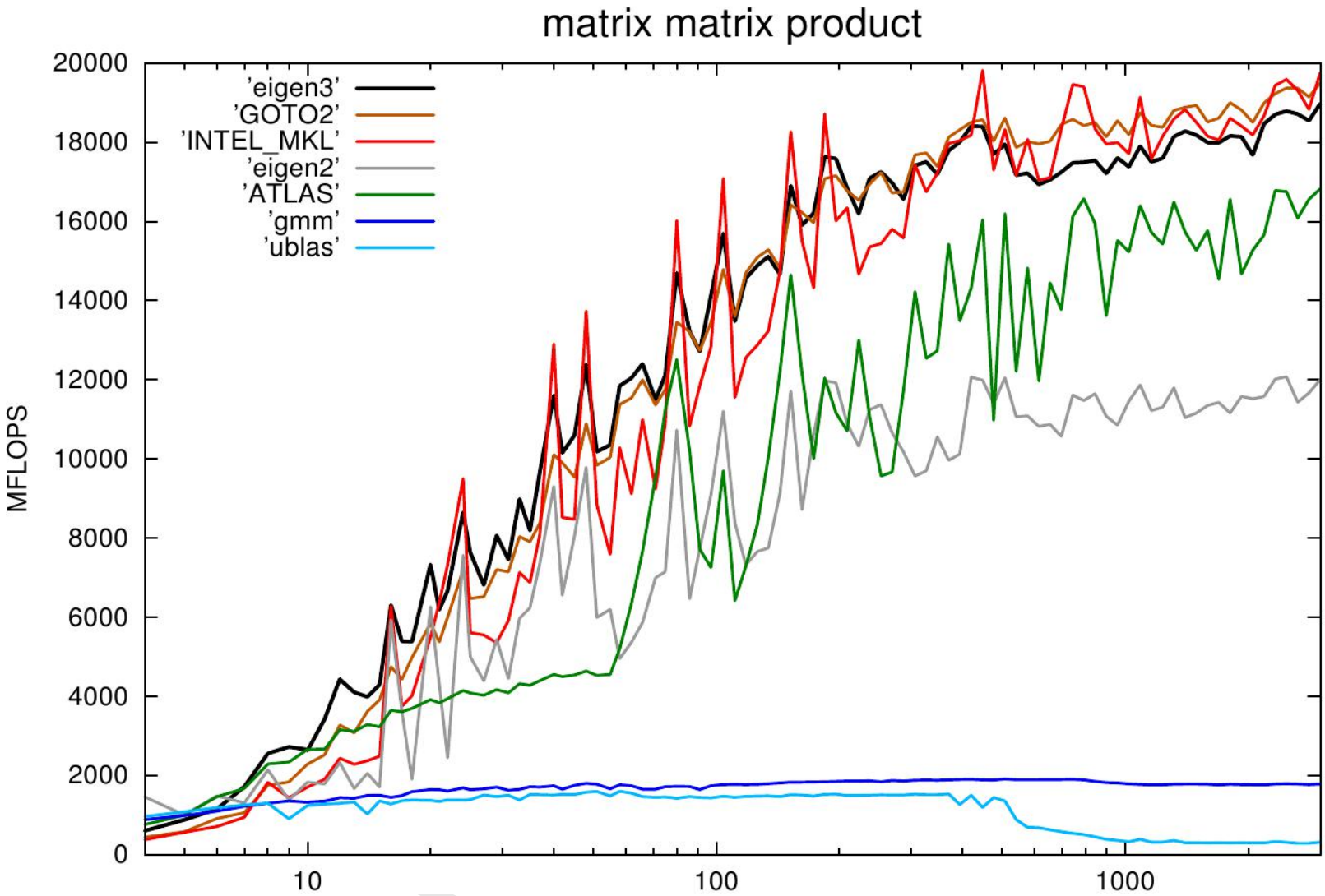
- Three levels of blocking:
  - L2/L3 → L1 → registers
  - A' and B' stored in a packed format
- Generic: scalar, # registers, conjugation, complex * real, etc.

Higher level products

- All routines are based on the unique GEBP kernel
  - ex, GEMM like routine:

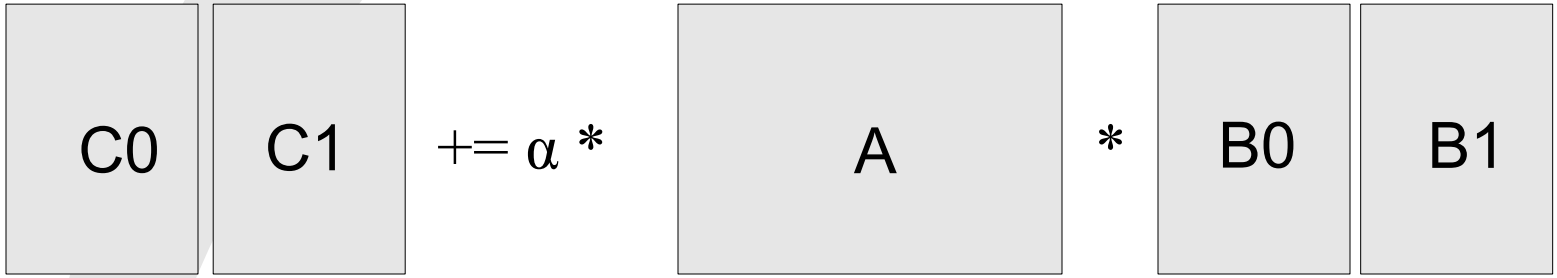


Performance

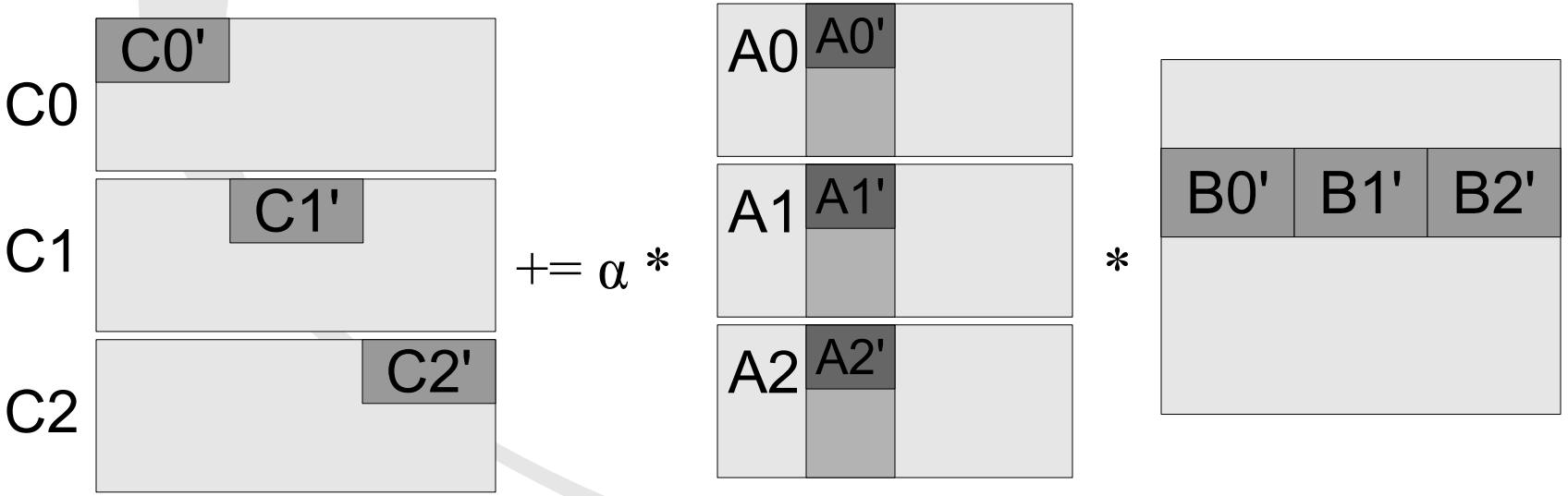


Parallelization (OpenMP)

- Naive approach:

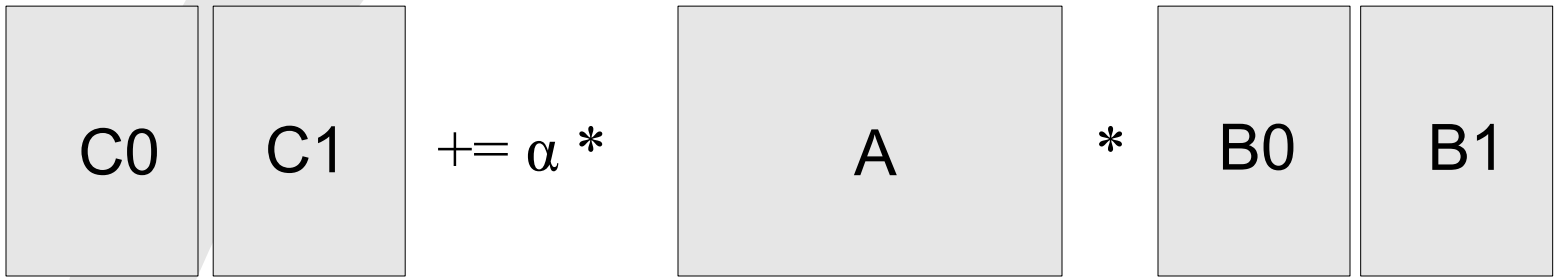


- Barrel shifter:

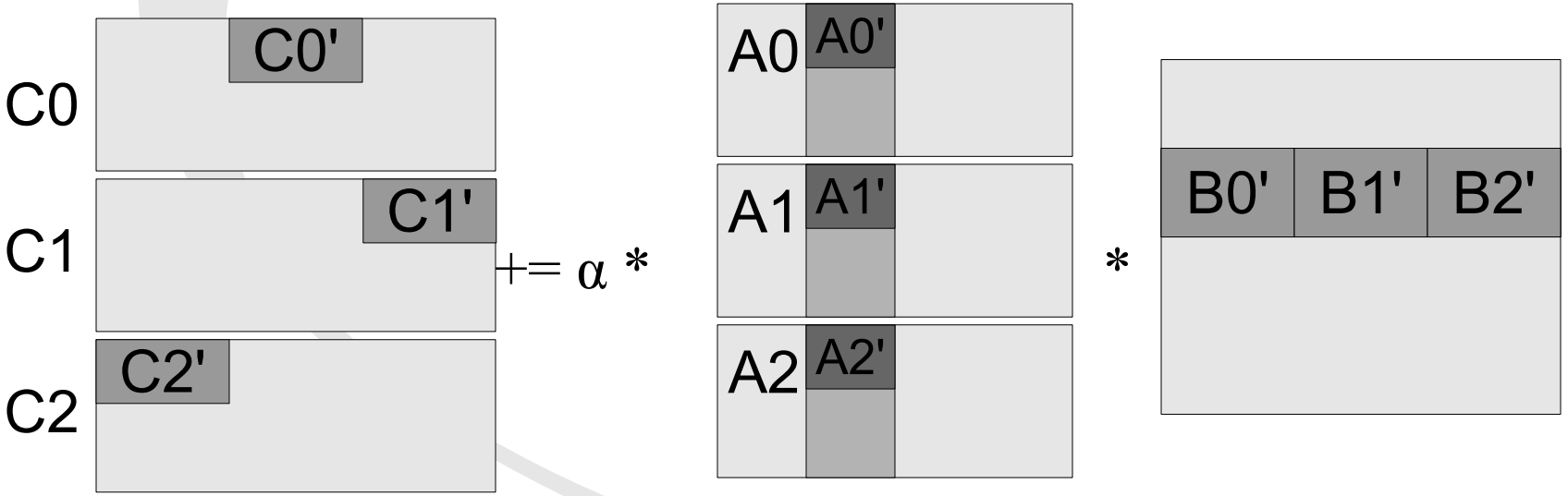


Parallelization (OpenMP)

- Naive approach:

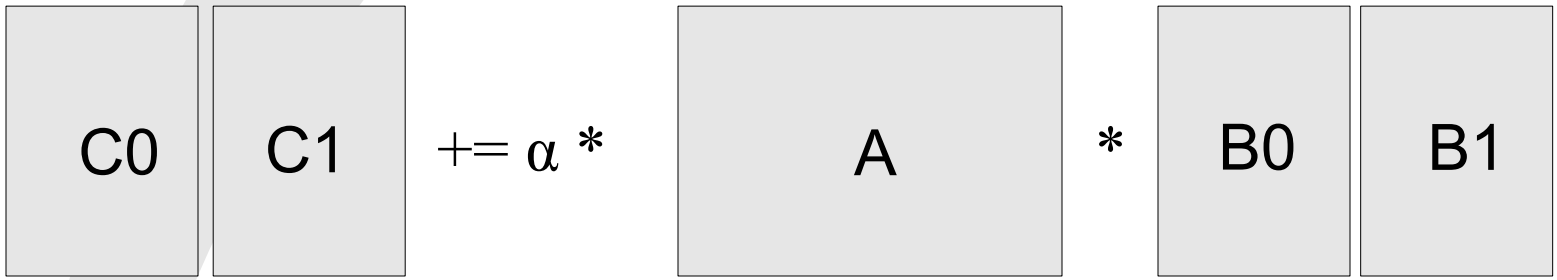


- Barrel shifter:

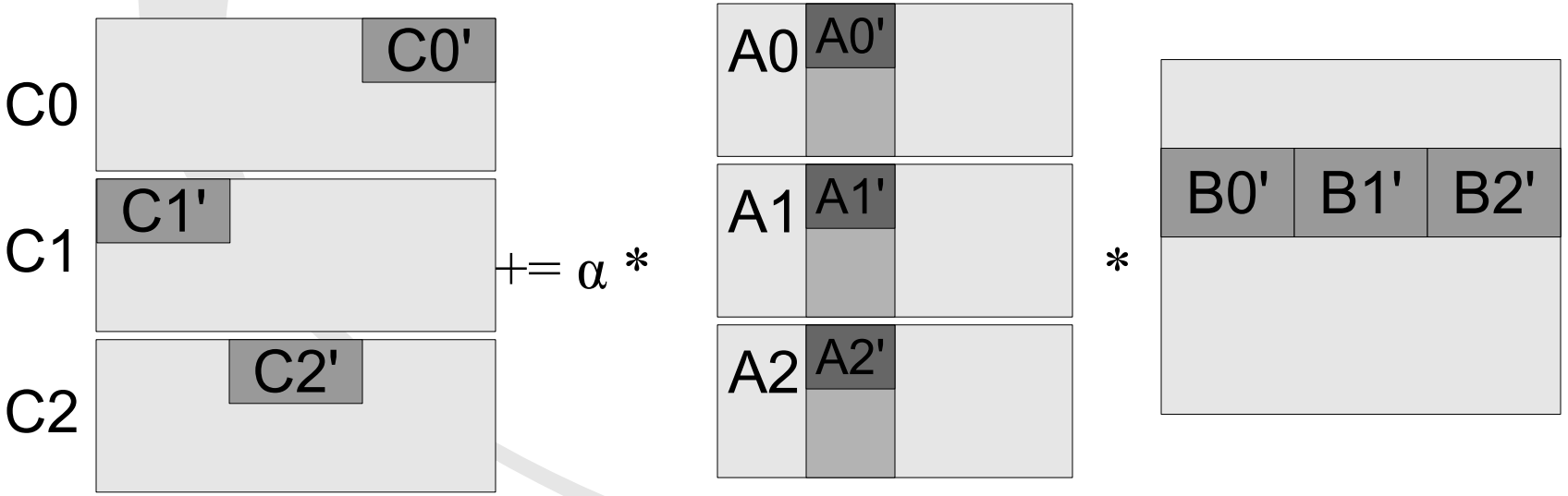


Parallelization (OpenMP)

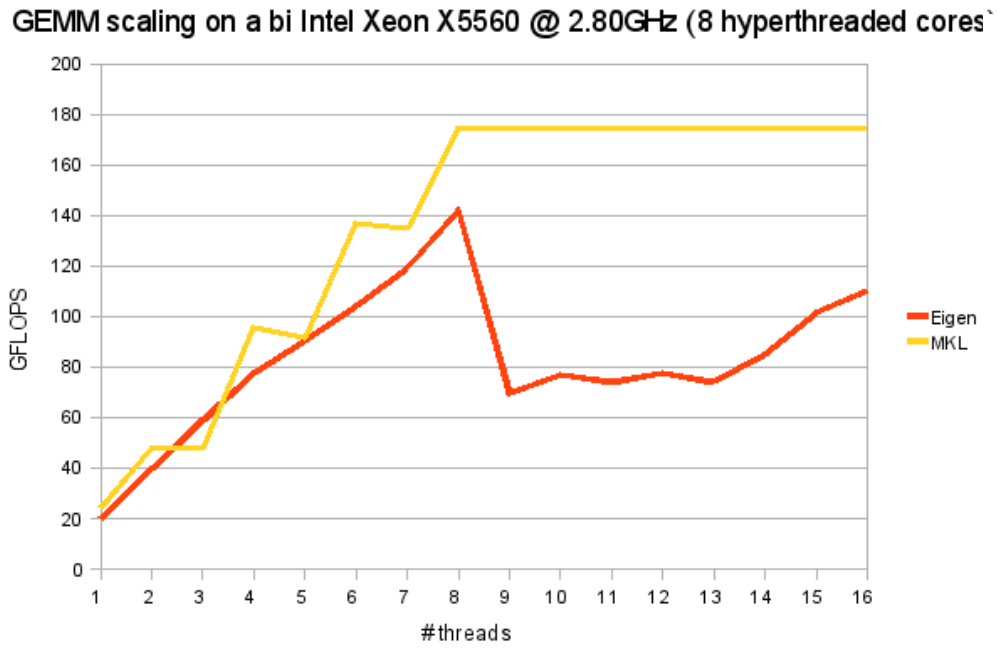
- Naive approach:

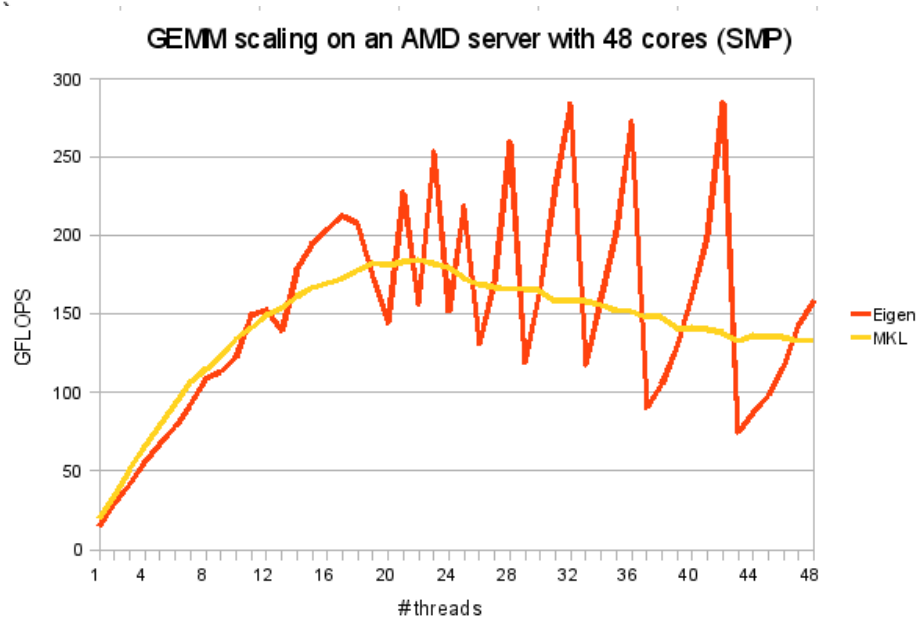


- Barrel shifter:



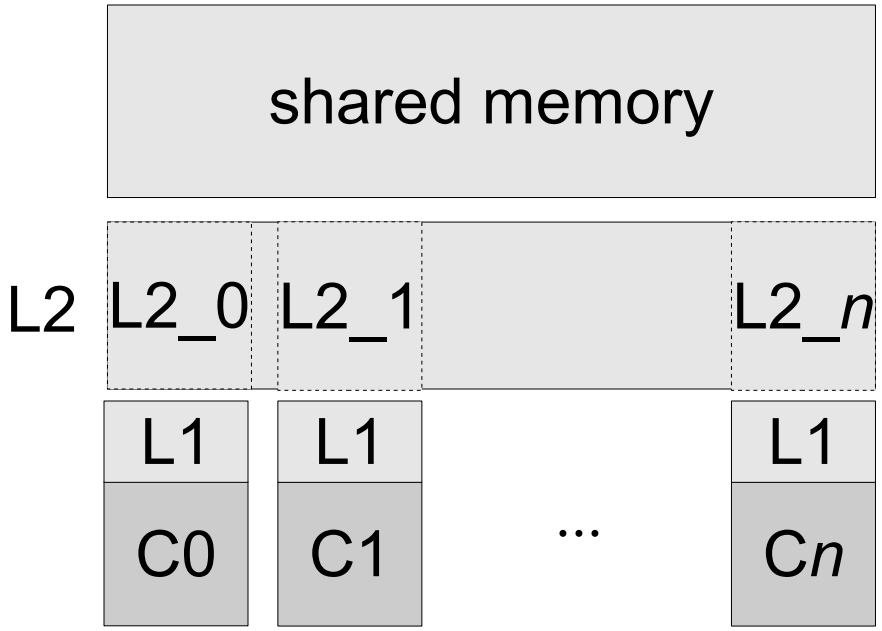
Parallelization : results





Products: summary

- Out-of-date SMP model:



→ MPI/OMP ?

- Generalize the approach to other products

Sparse Matrix (WIP)

- Current state
  - Compressed format
    - assembly, manipulation
  - Direct solver
    - Simplicial LIt/LDLt (AMD ordering)
      - todo: LU, supernodes, out-of-core, parallelization
    - Various backends (SuperLU,Umfpack,Cholmod, etc.)

```
1: typedef SparseMatrix<double,ColMajor> SpMat;
2: SpMat mat(rows,cols);
3: loop { mat.insert(i,j) = v_ij; }
4: SimplicialCholesky<SpMat,Lower> chol(mat);
5: x = chol.solve(b);
```

Sparse example

Expression templates

- Example:

```
m3 = m1 + m2 + m3;
```

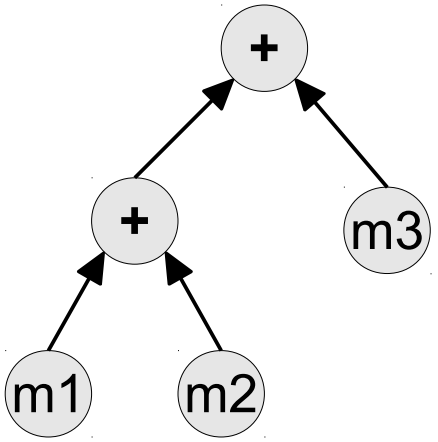
- Standard C++ way:

```
tmp1 = m1 + m2;
tmp2 = tmp1 + m3;
m3   = tmp2;
```

Expression templates

- Example:

```
m3 = m1 + m2 + m3;
```



- Expression templates:
  - "+" returns an expression
    => expression tree
  - e.g.: A+B returns:

```
Sum<type_of_A, type_of_B> {
    const type_of_A& A;
    const type_of_B& B;
};
```

  - complete example:

```
Assign<Matrix,
       Sum< Sum<Matrix,Matrix> , Matrix > >
```

Expression templates

- Example:

```
m3 = m1 + m2 + m3;
```

- Evaluation:
  - Top-down creation of an evaluator
    - e.g.:

```
Evaluator<Sum<type_of_A,type_of_B> > {
  Evaluator<type_of_A> evalA(A);
  Evaluator<type_of_B> evalB(B);
  Scalar coeff(i,j) {
    return evalA.coeff(i,j) + evalB.coeff(i,j);
  }
};
```

  - Assignment produces:

```
for(i=0; i<m3.size(); ++i)
    m3[i] = m1[i] + m2[i] + m3[i];
```

ET: Immediate benefits

- Temporary removal
- Reduce memory accesses
- Better API, ex:

```
x.col(4) = A.lu().solve(B.col(5));

x = b * A.triangularView<Lower>().inverse();
```

- Better unrolling
- Better vectorization

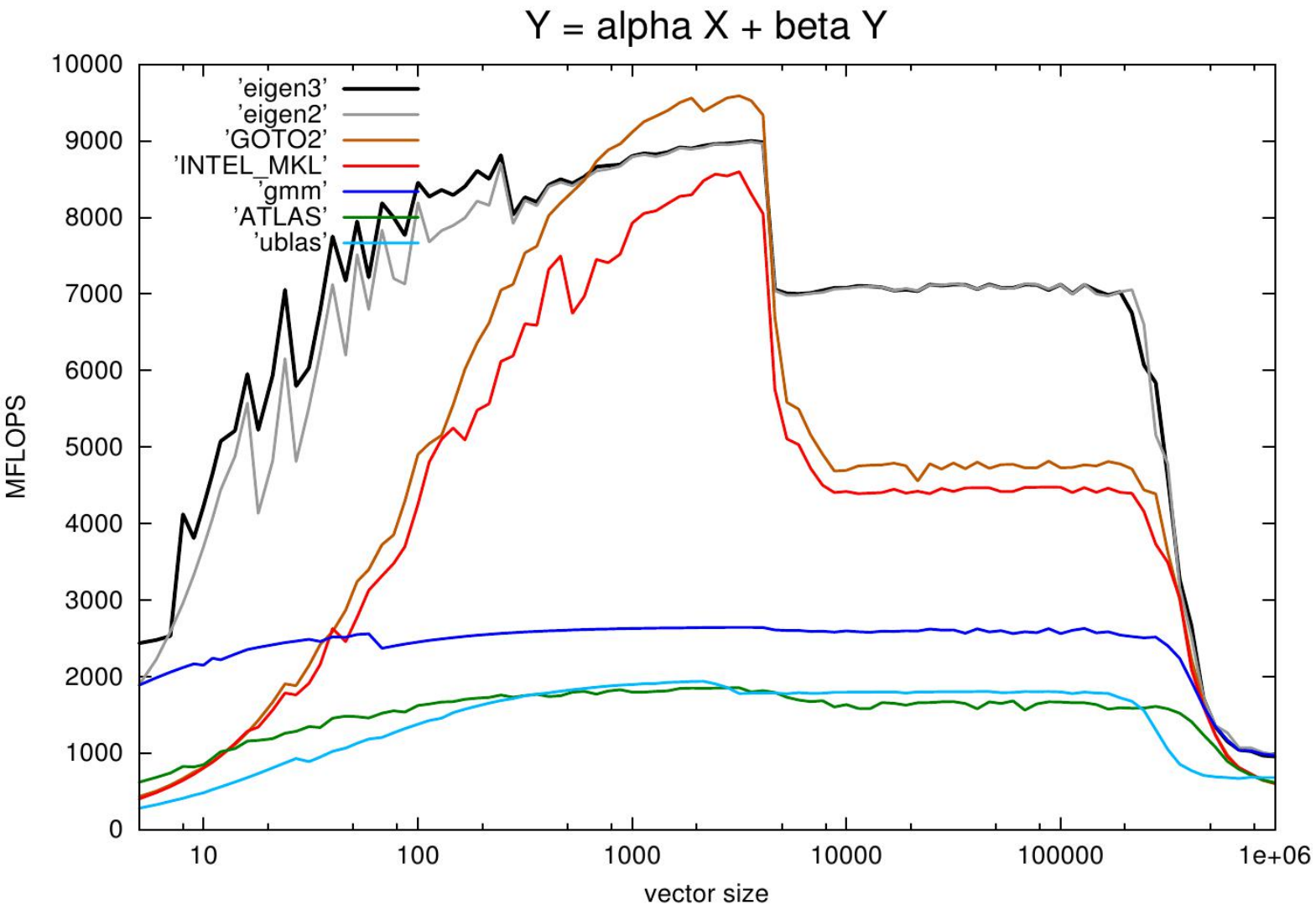
ET & Vectorization

```
#include<Eigen/Core>
using namespace Eigen;

void foo(Matrix2f& u,
         float a, const Matrix2f& v,
         float b, const Matrix2f& w)
{
  u = a*v + b*w - u;
}
```

```
movl 8(%ebp), %edx
movss 20(%ebp), %xmm0
movl 24(%ebp), %eax
movaps %xmm0, %xmm2
shufps $0, %xmm2, %xmm2
movss 12(%ebp), %xmm0
movaps %xmm2, %xmm1
mulps (%eax), %xmm1
shufps $0, %xmm0, %xmm0
movl 16(%ebp), %eax
mulps (%eax), %xmm0
addps %xmm1, %xmm0
subps (%edx), %xmm0
movaps %xmm0, (%edx)
```

Performance



Cost model

- Cost Model
  - Track an approximation of the cost to evaluate one coefficient
- Control of:
  - loop unrolling (partial)
  - evaluation of sub expressions, e.g.:
    - *(**a+b**) * c* → *(a+b)* is evaluated into a temporary
    - *enable vectorization of sub expressions (todo)*

Top-down expression analysis

- Products (again)
  - detect BLAS-like sub expressions
    - e.g.: `m4 -= 2 * m2.adjoint() * m3;`

      → `gemm<Adj,Nop>(m2, m3, -2, m4);`
    - e.g.: `m4.block(...) += ((2+4i) * m2).adjoint() * m3.block(...).transpose();`

```
Evaluator<Product<type_of_A,type_of_B> > {
  EvaluatorForProduct<type_of_A> evalA(A);
  EvaluatorForProduct<type_of_B> evalB(B);
};
```

Top-down expression analysis (cont.)

- More complex example:

```
m4 -= m1 + m2 * m3;
```

  - so far:

```
tmp = m2*m3;
m4 -= m1 + tmp;
```

  - better:

```
m4 -= m1;
m4 += m2 * m3;
```

```
// catch R = A + B * C
Evaluator<Assign<R,Sum<A,Product<B,C> > > { … };
```

Tree optimizer

- Even more complex example:

```
res -= m1 + m2 + m3*m4 + 2*m5 – m6*m7;
```

  - Tree optimizer

    →
```
res -= ((m1 + m2 + 2*m5) + m3*m4) + m6*m7;
```

  - yields:
```
res -= m1 + m2 + 2*m5;
res += m3*m4
res += m6*m7;
```

  - Need only two rules:

```
// catch A * B + Y and builds Y' + A' * B'
TreeOpt<Sum<Product<A,B>,Y> > { … };

// catch X + A * B + Y and builds (X' + Y') + (A' * B')
TreeOpt<Sum<Sum<X,Product<A,B> >,Y> >
```

Tree optimizer

- Last example:

```
res += m1 * m2 * v;
```

  - Tree optimizer

→

```
res += m1 * (m2 * v);
```

  - Rule:

```
TreeOpt<Product<Product<A,B>,C> > { … };
```

Conclusion

- Future
  - Full-time engineer from sept/oct (ADT)

! position still available !

  - Stabilization of many modules:
    - Sparse matrices, non linear optimization, etc.
  - Develop high level modules
    (Polynomial solver, linear regression, autodiff, …)
  - More parallelization, support for GPUs

Matrix decompositions

- LU: Full pivoting, Partial pivoting, Blocking
- Cholesky: LLT, LDLT, Pivoting, Blocking
- QR: Householder QR (No, column, full pivoting)
- SVD: Jacobi (Full or Thin, QR preconditionner)
  - Todo: Householder (faster for large matrices but less accurate)
- Eigenvalues:
  - self-adjoint:
    - householder triadiagonalization + QR iterations
    - optimizations for 2x2 and 3x3
  - General case: Householder Hessenberg + Shur decomposition (Francis QR iterations with implicit double shift)

Many users

- Robotics
  - **Willow Garage**: ROS, Point Cloud Lib, etc.
  - INRIA: e-Motion (*Manuel Yguel*)
  - MRPT
  - Yujin Robot
  - Darmstadt {Dribblers,Rescue Robot}
- Motion
  - Blender: iTaSC (constraint IK)
  - KDL (Kinematics & Dynamics Lib)
- **Google** (computer vision & machine learning)
- CEA, European Space Agency (space trajectory)
- Meshlab/vcglib, KDE (Krita,step), **etc.**

Good Documentation

- Generated every day
  - Doxygen doc with many examples
  - Tutorial/Manual
  - Quick reference guide
  - Quick MatLab to Eigen guide
  - Discussions on advanced topics

```
source code + inline doc      2.9M
user guide + snippets         1.6M
unit tests                    1.1M
```

Reliability

- Extensive unit tests
  - run every days on various platforms
  - include BLAS and Lapack test suites

My CDash | All Dashboards | Log Out

Friday, February 25 2011 07:56:38 UTC

EIGEN
Dashboard

DASHBOARD | CALENDAR | PREVIOUS | CURRENT | PROJECT | ADMINISTRATION

No update data as of Friday, February 25 2011 00:00:00 UTC

Help

[Show Filters]

Nightly

Site	Build Name	Update		Configure			Build			Test				Build Time	Labels
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min		
lfse	fedora-7-i686-gcc-4.1.2-SSE2	0	0	0	0	0	0	5	61.6	0	1	544	2	2011-02-25T02:43:50 UTC	(none)
lfse	fedora-7-i686-gcc-4.3.3-debug-SSE2	0	0	0	0	0	0	2	54.2	0	0	545	16.1	2011-02-25T04:46:49 UTC	(none)
lfse	fedora-7-i686-gcc-4.3.3-lowe	0	0	0	0	0	0	3	53.2	0	0	545	2.8	2011-02-25T05:58:23 UTC	(none)
lfse	fedora-7-i686-gcc-4.3.3-SSE2	0	0.1	0	0	0	0	4	56	0	1	544	1	2011-02-25T03:48:38 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-3.4.6	0	0	0	0	0	0	51	37.1	0	16	518	2.8	2011-02-25T03:23:37 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.0.1	0	0	0	0	0	0	9	35	0	0	545	2	2011-02-25T02:45:33 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.2.4-SSE2	0	0	0	0	0	0	39	36	0	0	548	0.9	2011-02-25T04:04:38 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.3.4-SSE2	0	0	0	0	0.1	0	4	27.2	0	0	548	1	2011-02-25T02:16:16 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.5.0-dbg-SSE2	0	0	0	0	0	0	5	30.6	0	0	548	17.7	2011-02-25T00:00:39 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.5.0-openmp-SSE4_1	0	0	0	0	0	0	5	19.6	0	1	547	0.7	2011-02-25T01:13:21 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.5.0-rel-SSE2	0	0	0	0	0	0	5	19.2	0	2	545	0.9	2011-02-25T00:51:47 UTC	(none)
po-qael	opensuse-11.1-x86_64-gcc-4.5.0-rowmaj-SSE3	0	0	0	0	0	0	5	38.9	0	0	548	1.3	2011-02-25T01:34:22 UTC	(none)
Totals	12 Builds	0	0.1	0	0	0.1	0	137	468.6	0	21	6527	49.2		

No Continuous Builds

Experimental

Site	Build Name	Update		Configure			Build			Test				Build Time	Labels
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min		
GORDO	windows-vista-x86_32-MSVC-2010/10.0.30319.1-SSE2			0	0	0.4	0	265	244.9	0	0	532	4.2	2011-02-25T01:02:24 UTC	(none)