

# The Tensor Contraction Engine: Automated Program Synthesis for a Class of Quantum Chemistry Computations

Oak Ridge National  
Laboratory

*David E. Bernholdt*  
*Robert Harrison*

Pacific Northwest National  
Laboratory

*Jarek Nieplocha*

Louisiana State University

*Gerald Baumgartner*  
*J. Ramanujam*

Ohio State University

*Xiaoyang Gao*  
*Albert Hartono*  
*Sriram Krishnamoorthy*  
*Qingda Lu*  
*Russell Pitzer*  
*P. (Saday) Sadayappan*  
*Chiranjib Sur*

University of Florida  
So Hirata

University of Waterloo  
Marcel Nooijen

---

Supported by NSF and DOE

# Tensor Contraction Engine

- Automatic transformation from high-level specification
  - Chemist specifies computation in high-level mathematical form
  - Synthesis system transforms it to efficient parallel program
  - Code is tailored to target machine
  - Code can be optimized for specific molecules being modeled
- Multi-institutional collaboration (OSU, LSU, Waterloo, ORNL, PNNL, U. Florida)
- Two prototypes of TCE are operational
  - a) Full exploitation of symmetry, but fewer optimizations (So Hirata), b) Dense tensors, but more sophisticated optimizations; Integrated version with all optimizations and exploitation of symmetry is nearing completion
  - Has been used to implement over 20 models, included in latest release of NWChem
  - First parallel implementation for many of the methods
  - TCE Workshop at Sanibel 2007

$$A3A = \frac{1}{2} (X_{ce,af} Y_{ae,cf} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},af} Y_{ae,\bar{c}\bar{f}} + X_{ce,\bar{a}\bar{f}} Y_{ae,\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{ce,\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}})$$
$$X_{ce,af} = t_{ij}^{ce} t_{ij}^{af} \quad Y_{ae,cf} = \langle ab \| ek \rangle \langle cb \| fk \rangle$$

```
range V = 3000;
range O = 100;

index a,b,c,d,e,f : V;
index i,j,k : O;

mlimit = 10000000;

function F1(V,V,V,O);
function F2(V,V,V,O);

procedure P(in T1[O,O,V,V], in T2[O,O,V,V], out X)=

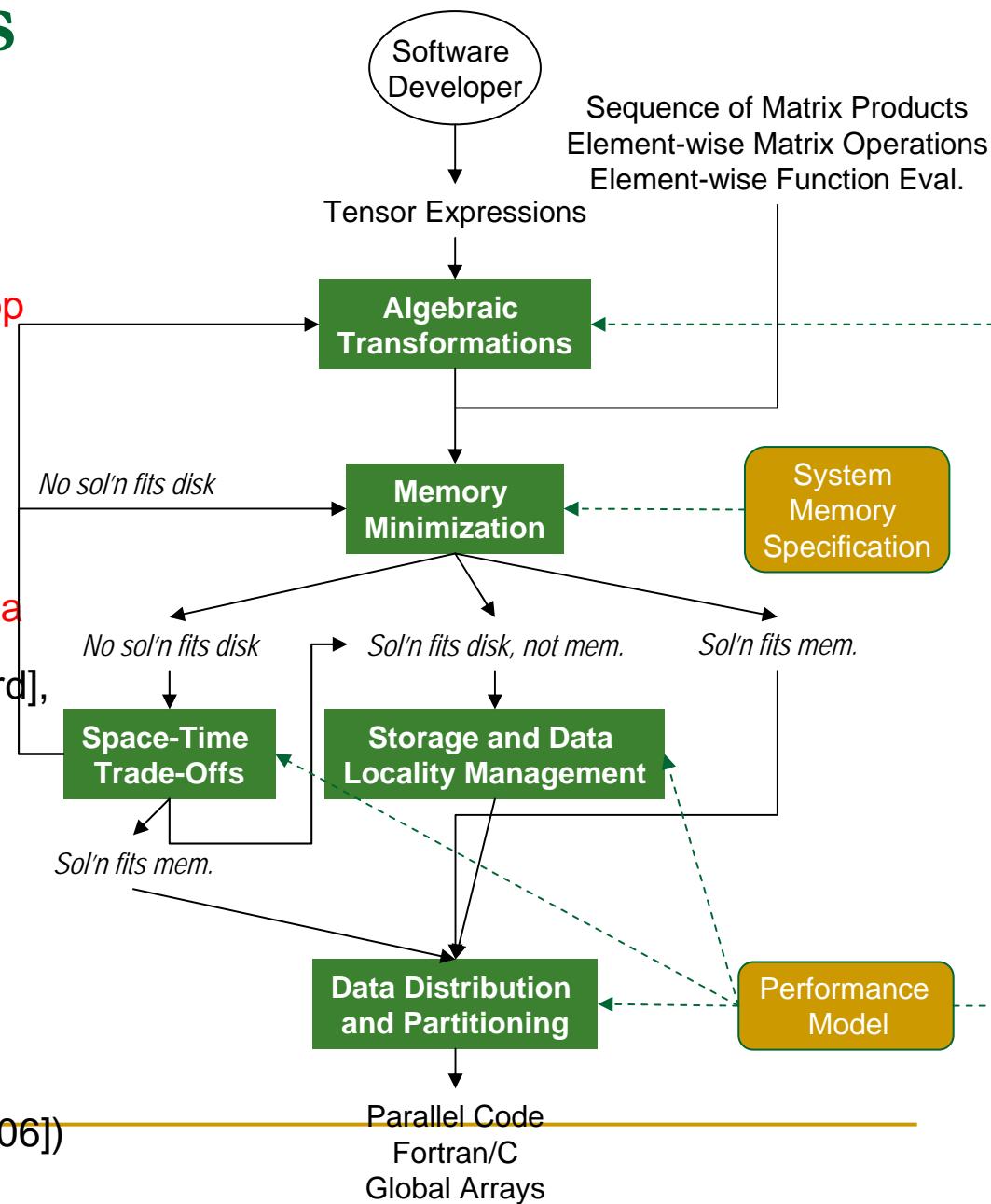
begin
    A3A == sum[ sum[ F1(a,b,e,k) * F2(c,f,b,k), {b,k} ]
                * sum[ T1[i,j,c,e] * T2[i,j,a,f], {i,j} ],
                {a,e,c,f}] * 0.5 + ...;
end
```

# CCSD Doubles Equation (Quantum Chemist's Eye Test Chart :-)

```
hbar[a,b,i,j] == sum[f[b,c]*t[i,j,a,c],{c}] -sum[f[k,c]*t[k,b]*t[i,j,a,c],{k,c}] +sum[f[a,c]*t[i,j,c,b],{c}] -sum[f[k,c]*t[k,a]*t[i,j,c,b],{k,c}] -  
sum[f[k,j]*t[i,k,a,b],{k}] -sum[f[k,c]*t[j,c]*t[i,k,a,b],{k,c}] -sum[f[k,i]*t[j,k,b,a],{k}] -sum[f[k,c]*t[i,c]*t[j,k,b,a],{k,c}]  
+sum[t[i,c]*t[j,d]*v[a,b,c,d],{c,d}] +sum[t[i,j,c,d]*v[a,b,c,d],{c,d}] +sum[t[j,c]*v[a,b,i,c],{c}] -sum[t[k,b]*v[a,k,i,j],{k}] +sum[t[i,c]*v[b,a,j,c],{c}]  
-sum[t[k,a]*v[b,k,j,i],{k}] -sum[t[k,d]*t[i,j,c,b]*v[k,a,c,d],{k,c,d}] -sum[t[i,c]*t[j,k,b,d]*v[k,a,c,d],{k,c,d}] -sum[t[j,c]*t[k,b]*v[k,a,c,i],{k,c}]  
+2*sum[t[j,k,b,c]*v[k,a,c,i],{k,c}] -sum[t[i,c]*t[j,d]*v[k,b]*v[k,a,d,c],{k,c,d}]  
+2*sum[t[k,d]*t[i,j,c,b]*v[k,a,d,c],{k,c,d}] -sum[t[k,b]*t[i,j,c,d]*v[k,a,d,c],{k,c,d}] -sum[t[j,d]*t[i,k,c,b]*v[k,a,d,c],{k,c,d}]  
+2*sum[t[i,c]*t[j,k,b,d]*v[k,a,d,c],{k,c,d}] -sum[t[i,c]*t[j,k,d,b]*v[k,a,d,c],{k,c,d}] -sum[t[j,k,b,c]*v[k,a,i,c],{k,c}] -  
sum[t[i,c]*t[k,b]*v[k,a,j,c],{k,c}] -sum[t[i,k,c,b]*v[k,a,j,c],{k,c}] -sum[t[i,c]*t[j,d]*t[k,a]*v[k,b,c,d],{k,c,d}] -  
sum[t[k,d]*t[i,j,a,c]*v[k,b,c,d],{k,c,d}] -sum[t[k,a]*t[i,j,c,d]*v[k,b,c,d],{k,c,d}] +2*sum[t[j,d]*t[i,k,a,c]*v[k,b,c,d],{k,c,d}] -  
sum[t[j,d]*t[i,k,c,a]*v[k,b,c,d],{k,c,d}] -sum[t[i,c]*t[j,k,d,a]*v[k,b,c,d],{k,c,d}] -sum[t[i,c]*t[k,a]*v[k,b,c,j],{k,c}]  
+2*sum[t[i,k,a,c]*v[k,b,c,j],{k,c}] -sum[t[i,k,c,a]*v[k,b,c,j],{k,c}] +2*sum[t[k,d]*t[i,j,a,c]*v[k,b,d,c],{k,c,d}] -  
sum[t[j,d]*t[i,k,a,c]*v[k,b,d,c],{k,c,d}] -sum[t[j,c]*t[k,a]*v[k,b,i,c],{k,c}] -sum[t[j,k,c,a]*v[k,b,i,c],{k,c}] -sum[t[i,k,a,c]*v[k,b,j,c],{k,c}]  
+sum[t[i,c]*t[j,d]*t[k,a]*t[l,b]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[k,b]*t[l,d]*t[i,j,a,c]*v[k,l,c,d],{k,l,c,d}] -  
2*sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[k,a]*t[l,b]*t[i,j,c,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[j,c]*t[l,d]*t[i,k,a,b]*v[k,l,c,d],{k,l,c,d}] -  
2*sum[t[j,d]*t[l,b]*t[i,k,a,c]*v[k,l,c,d],{k,l,c,d}] +sum[t[j,d]*t[l,b]*t[i,k,c,a]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,c]*t[l,d]*t[j,k,b,a]*v[k,l,c,d],{k,l,c,d}]  
+sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,b]*t[j,k,d,a]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,c,d],{k,l,c,d}] -  
4*sum[t[i,k,a,c]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,a,b]*t[j,l,c,d]*v[k,l,c,d],{k,l,c,d}] -  
2*sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,k,c,a]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,c]*t[j,d]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}]  
+sum[t[i,j,c,d]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,j,c,b]*t[k,l,a,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,j,a,c]*t[k,l,b,d]*v[k,l,c,d],{k,l,c,d}]  
+sum[t[j,c]*t[k,b]*t[l,a]*v[k,l,c,i],{k,l,c}] +sum[t[l,c]*t[j,k,b,a]*v[k,l,c,i],{k,l,c}] -2*sum[t[l,a]*t[j,k,b,c]*v[k,l,c,i],{k,l,c}]  
+sum[t[l,a]*t[j,k,c,b]*v[k,l,c,i],{k,l,c}] -2*sum[t[k,c]*t[j,l,b,a]*v[k,l,c,i],{k,l,c}] +sum[t[k,a]*t[j,l,b,c]*v[k,l,c,i],{k,l,c}]  
+sum[t[k,b]*t[j,l,c,a]*v[k,l,c,i],{k,l,c}] +sum[t[j,c]*t[l,k,a,b]*v[k,l,c,i],{k,l,c}] +sum[t[i,c]*t[k,a]*t[l,b]*v[k,l,c,j],{k,l,c}]  
+sum[t[l,c]*t[i,k,a,b]*v[k,l,c,j],{k,l,c}] -2*sum[t[l,b]*t[i,k,a,c]*v[k,l,c,j],{k,l,c}] +sum[t[l,b]*t[i,k,c,a]*v[k,l,c,j],{k,l,c}]  
+sum[t[i,c]*t[k,l,a,b]*v[k,l,c,j],{k,l,c}] +sum[t[j,c]*t[l,d]*t[i,k,a,b]*v[k,l,d,c],{k,l,c,d}] +sum[t[j,d]*t[l,b]*t[i,k,a,c]*v[k,l,d,c],{k,l,c,d}]  
+sum[t[j,d]*t[l,a]*t[i,k,c,b]*v[k,l,d,c],{k,l,c,d}] -2*sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,d,c],{k,l,c,d}] -2*sum[t[i,k,a,c]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}]  
+sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,a,b]*t[j,l,c,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,c,b]*t[j,l,d,a]*v[k,l,d,c],{k,l,c,d}]  
+sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,d,c],{k,l,c,d}] +sum[t[k,a]*t[l,b]*v[k,l,i,j],{k,l}] +sum[t[k,l,a,b]*v[k,l,i,j],{k,l}]  
+sum[t[k,b]*t[l,d]*t[i,j,a,c]*v[l,k,c,d],{k,l,c,d}] +sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,d]*t[j,k,b,a]*v[l,k,c,d],{k,l,c,d}] -  
2*sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,a]*t[j,k,d,b]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,j,c,b]*t[k,l,a,d]*v[l,k,c,d],{k,l,c,d}]  
+sum[t[i,j,a,c]*t[k,l,b,d]*v[l,k,c,d],{k,l,c,d}] -2*sum[t[l,c]*t[i,k,a,b]*v[l,k,c,j],{k,l,c}] +sum[t[l,b]*t[i,k,a,c]*v[l,k,c,j],{k,l,c}]  
+sum[t[l,a]*t[i,k,c,b]*v[l,k,c,j],{k,l,c}] +v[a,b,i,j]
```

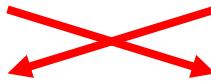
# TCE Components

- Algebraic Transformations
  - Minimize operations (ICCS'05, ICCS'06)
- Memory Minimization
  - Reduce intermediate storage via loop fusion (LCPC'03)
- Space-Time Transformation
  - Trade-offs between storage and recomputation (PLDI'02)
- Data Locality Optimization
  - Optimize use of storage hierarchy via tiling (ICS'01, HiPC'03 [Best Paper Award], IPDPS'04 [Best Paper Award], IPDPS'05)
- Data Dist./Comm. Optimization
  - Optimize data layout/distribution & computation partitioning (IPDPS'03, LCPC'04, PPoPP'05, SC05)
- Runtime Framework
  - Locality-Conscious Load-Balancing (SC06)
- Integrated System
  - (SC'02, Proc. IEEE '05, Mol. Phys. '06)



# Algebraic Transformations for Op-Min

$$S(a,b,i,j) = \sum_{c,d,e,f,k,l} A(a,c,i,k) B(b,e,f,l) C(d,f,j,k) D(c,d,e,l)$$



$$S(a,b,i,j) = \sum_{c,d,e,f,k,l} A(a,c,i,k) C(d,f,j,k) B(b,e,f,l) D(c,d,e,l)$$

$$S(a,b,i,j) = \sum_{c,d,f,k} A(a,c,i,k) C(d,f,j,k) \left( \sum_{e,l} B(b,e,f,l) D(c,d,e,l) \right)$$

$$S(a,b,i,j) = \sum_{c,k} A(a,c,i,k) \left[ \sum_{d,f} C(d,f,j,k) \left( \sum_{e,l} B(b,e,f,l) D(c,d,e,l) \right) \right]$$

---

$$T1(b,c,d,f) = \sum_{e,l} B(b,e,f,l) D(c,d,e,l) \quad 2N^6 \text{ Ops}$$

$$T2(b,c,j,k) = \sum_{d,f} T1(b,c,d,f) C(d,f,j,k) \quad 2N^6 \text{ Ops}$$

$$S(a,b,i,j) = \sum_{c,k} T2(b,c,j,k) A(a,c,i,k) \quad 2N^6 \text{ Ops}$$

---

# Algebraic Transformations: Operation Minimization

$$S(a, b, i, j) = \sum_{c,d,e,f,k,l} A(a, c, i, k) B(b, e, f, l) C(d, f, j, k) D(c, d, e, l)$$

- Requires  $4 * N^{10}$  operations if indices  $a-l$  have range  $N$
- Optimized form requires only  $6 * N^6$  operations

$$T1(b, c, d, f) = \sum_{e, l} B(b, e, f, l) D(c, d, e, l)$$

$$T2(b, c, j, k) = \sum_{d, f} T1(b, c, d, f) C(d, f, j, k)$$

$$S(a, b, i, j) = \sum_{c, k} T2(b, c, j, k) A(a, c, i, k)$$

- Optimization Problem: Given an input tensor-contraction expression, find equivalent form that minimizes # operations
  - Problem is NP-hard; efficient pruning search strategy developed, that appears to be effective in practice
- However, storage reqmts. increase after operation minimization

# Memory Minimization: Compute by Parts (Loop Fusion)

$$\begin{aligned} T1_{bcdf} &= \sum_{e,l} B_{befl} D_{cdel} \\ T2_{bcjk} &= \sum_{d,f} T1_{bcdf} C_{dfjk} \\ S_{abij} &= \sum_{c,k} T2_{bcjk} A_{acik} \end{aligned}$$

Formula sequence

T1 = 0; T2 = 0; S = 0  
for **b, c, d, e, f, l**  
  └ T1<sub>bcdf</sub> += B<sub>befl</sub> D<sub>cdel</sub>  
    for **b, c, d, f, j, k**  
      └ T2<sub>bcjk</sub> += T1<sub>bcdf</sub> C<sub>dfjk</sub>  
        for **a, b, c, i, j, k**  
          └ S<sub>abij</sub> += T2<sub>bcjk</sub> A<sub>acik</sub>

Unfused code

S = 0  
for **b, c**  
  └ T1f = 0; T2f = 0  
    for **d, e, f, l**  
      └ T1f<sub>df</sub> += B<sub>befl</sub> D<sub>cdel</sub>  
        for **d, f, j, k**  
          └ T2f<sub>jk</sub> += T1f<sub>df</sub> C<sub>dfjk</sub>  
            for **a, i, j, k**  
              └ S<sub>abij</sub> += T2f<sub>jk</sub> A<sub>acik</sub>

(Partially) Fused code

# Memory Minimization: Loop Fusion

$T1 = 0; T2 = 0; S = 0$

for b, c, d, e, f, l

$T1_{bcdf} += B_{befl} D_{cdel}$

for b, c, d, f, j, k

$T2_{bcjk} += T1_{bcdf} C_{dfjk}$

for a, b, c, i, j, k

$S_{abij} += T2_{bcjk} A_{acik}$

Unfused code

$S = 0$

for b, c

$T1f = 0; T2f = 0$

    for d, e, f, l

$T1f_{df} += B_{befl} D_{cdel}$

    for d, f, j, k

$T2f_{jk} += T1f_{df} C_{dfjk}$

    for a, i, j, k

$S_{abij} += T2f_{jk} A_{acik}$

(Partially) Fused code

$S = 0$

for b, c

$T1f = 0; T2f = 0$

  for d, f

    for e, l

$T1f += B_{befl} D_{cdel}$

    for j, k

$T2f_{jk} += T1f C_{dfjk}$

    for a, i, j, k

$S_{abij} += T2f_{jk} A_{acik}$

Fully Fused code

- Optimization Problem: Given an operation-minimized sequence of tensor-contractions, find “best” set of loops to fuse
  - Problem is believed to be NP-hard; Heuristics and pruning search used

# An Operation-Minimization Experiment

- Can mechanical search automatically produce “new” formulations, similar to AO-based forms?
- Explicitly expand out expressions for AO-to-MO transformation and feed it with CCSD T1/T2 equations to Op-Min module
- How does the generated form compare with the standard form?

# Experiment: Standard CCSD T1

MO integrals

AO to MO

AO integrals

$$\begin{aligned}
 v\_ooov^{h1h2}_{h3p1} &= (c\_mo^{q1}_{h3} * c\_mv^{q2}_{p1} * c\_om^{h1}_{q3} * c\_om^{h2}_{q4} * a\_mmmm^{q3q4}_{q1q2}) \\
 v\_oovv^{h1h2}_{p1p2} &= (c\_mv^{q1}_{p1} * c\_mv^{q2}_{p2} * c\_om^{h1}_{q3} * c\_om^{h2}_{q4} * a\_mmmm^{q3q4}_{q1q2}) \\
 v\_ovov^{h1p1}_{h2p2} &= (c\_mo^{q1}_{h2} * c\_mv^{q2}_{p2} * c\_om^{h1}_{q3} * c\_vm^{p1}_{q4} * a\_mmmm^{q3q4}_{q1q2}) \\
 v\_ovvv^{h1p1}_{p2p3} &= (c\_mv^{q1}_{p2} * c\_mv^{q2}_{p3} * c\_om^{h1}_{q3} * c\_vm^{p1}_{q4} * a\_mmmm^{q3q4}_{q1q2})
 \end{aligned}$$

$$\begin{aligned}
 \text{residual}_{h1}^{p2} = & 0.25 * (t\_vvo^{p2p1}_{h2h1} * f\_ov^{h2}_{p2}) - 0.25 * (v\_ovov^{h2p1}_{h1p2} * t\_vo^{p2}_{h2}) \\
 & + 0.25 * (f\_vv^{p1}_{p2} * t\_vo^{p2}_{h1}) - 0.25 * (f\_oo^{h2}_{h1} * t\_vo^{p1}_{h2}) + 0.25 * f\_vo^{p1}_{h1} \\
 & - 0.25 * (t\_vo^{p1}_{h2} * t\_vo^{p2}_{h1} * t\_vo^{p3}_{h3} * v\_oovv^{h2h3}_{p2p3}) \\
 & + 0.25 * (t\_vvoo^{p2p1}_{h2h1} * t\_vo^{p3}_{h3} * v\_oovv^{h2h3}_{p2p3}) - 0.125 * (t\_vo^{p1}_{h2} * t\_vvoo^{p2p3}_{h3h1} * v\_oovv^{h3h2}_{p2p3}) \\
 & - 0.125 * (t\_vo^{p2}_{h1} * t\_vvoo^{p3p1}_{h2h3} * v\_oovv^{h2h3}_{p3p2}) - 0.25 * (t\_vo^{p2}_{h1} * v\_ovvv^{h2p1}_{p2p3} * t\_vo^{p3}_{h2}) \\
 & - 0.25 * (t\_vo^{p1}_{h2} * v\_oooov^{h2h3}_{h1p2} * t\_vo^{p2}_{h3}) - 0.25 * (t\_vo^{p1}_{h2} * t\_vo^{p2}_{h1} * f\_ov^{h2}_{p2}) \\
 & + 0.125 * (t\_vvoo^{p2p3}_{h2h1} * v\_ovvv^{h2p1}_{p2p3}) + 0.125 * (t\_vvoo^{p2p1}_{h2h3} * v\_oooov^{h2h3}_{h1p2})
 \end{aligned}$$

# Experiment: Optimized CCSD T1

$$\begin{aligned}
it\_1^{q1h1}_{q2q3} &= (a\_mmmm_{q2q3}^{q4q1} * c\_om_{q4}^{h1}) \\
it\_2^{h1h2}_{p1q1} &= (c\_mv_{p1}^{q2} * (c\_om_{q3}^{h1} * it\_1^{q3h2}_{q1q2})) \\
v\_oovv_{p1p2}^{h1h2} &= (c\_mv_{p1}^{q1} * it\_2^{h2h1}_{p2q1}) \\
it\_4^{h1h2}_{h3p1} &= (c\_mo_{h3}^{q1} * it\_2^{h1h2}_{p1q1}) \\
it\_3^{q1}_{h1} &= (c\_mv_{p1}^{q1} * t\_vo_{h1}^{p1}) \\
it\_5^{q1}_{q2} &= (it\_1^{q1h1}_{q2q3} * it\_3^{q3}_{h1}) \\
it\_6^{h1}_{p1} &= (v\_oovv_{p1p2}^{h1h2} * t\_vo_{h2}^{p2}) \\
\text{residual}_{h1}^{p2} &= 0.25 * f\_vo_{h1}^{p2} - 0.25 * (f\_oo_{h1}^{h2} * t\_vo_{h2}^{p1}) + 0.25 * (f\_vv_{p2}^{p1} * t\_vo_{h1}^{p2}) \\
&\quad + 0.125 * (c\_vm_{q1}^{p1} * (it\_1^{q1h2}_{q2q3} * (c\_mv_{p1}^{q1} * (c\_mv_{p2}^{q1} * t\_vvvv_{h1h2}^{p2p1})))) \\
&\quad - 0.25 * ((f\_ov_{p1}^{h1} * t\_vo_{h2}^{p1}) * t\_vo_{h2}^{p1}) - 0.125 * ((t\_vo_{h3}^{p2} * v\_oo_{p1p2}^{h1h2}) * t\_vvvv_{h2h3}^{p2p1}) \\
&\quad - 0.125 * ((t\_vvvv_{h3h2}^{p1p2} * v\_oo_{p1p2}^{h3h1}) * t\_vo_{h2}^{p1}) + 0.25 * (t\_vvvv_{h2h1}^{p2p1} * it\_6^{h2}_{p2}) \\
&\quad - 0.25 * ((it\_6^{h1}_{p1} * t\_vo_{h2}^{p1}) * t\_vo_{h2}^{p1}) - 0.25 * (c\_vm_{q1}^{p1} * (c\_mo_{h1}^{q2} * it\_5^{q1}_{q2})) \\
&\quad - 0.25 * (c\_vm_{q1}^{p1} * (it\_3^{q2}_{h1} * it\_5^{q1}_{q2})) + 0.125 * (it\_4^{h2h3}_{h1p2} * t\_vvvv_{h3h2}^{p2p1}) \\
&\quad - 0.25 * ((it\_4^{h3h1}_{h2p1} * t\_vo_{h3}^{p1}) * t\_vo_{h2}^{p1}) + 0.25 * (t\_vvvv_{h2h1}^{p2p1} * f\_ov_{p2}^{h2})
\end{aligned}$$

# Experiment: Standard CCSD T1

$$v\_ooov^{h1h2}_{h3p1} = (c\_mo^{q1}_{h3} * c\_mv^{q2}_{p1} * c\_om^{h1}_{q3} * c\_om^{h2}_{q4} * a\_mmmm^{q3q4}_{q1q2})$$

$$v\_oovv^{h1h2}_{p1p2} = (c\_mv^{q1}_{p1} * c\_mv^{q2}_{p2} * c\_om^{h1}_{q3} * c\_om^{h2}_{q4} * a\_mmmm^{q3q4}_{q1q2})$$

$$v\_ovov^{h1p1}_{h2p2} = (c\_mo^{q1}_{h2} * c\_mv^{q2}_{p2} * c\_om^{h1}_{q3} * c\_vm^{p1}_{q4} * a\_mmmm^{q3q4}_{q1q2})$$

$$v\_ovvv^{h1p1}_{p2p3} = (c\_mv^{q1}_{p2} * c\_mv^{q2}_{p3} * c\_om^{h1}_{q3} * c\_vm^{p1}_{q4} * a\_mmmm^{q3q4}_{q1q2})$$

$$\begin{aligned} \text{residual}^{p2}_{h1} = & 0.25 * (t\_vvoo^{p2p1}_{h2h1} * f\_ov^{h2}_{p2}) - 0.25 * (v\_ovov^{h2p1}_{h1p2} * t\_vo^{p2}_{h2}) \\ & + 0.25 * (f\_vv^{p1}_{p2} * t\_vo^{p2}_{h1}) - 0.25 * (f\_oo^{h2}_{h1} * t\_vo^{p1}_{h2}) + 0.25 * f\_vo^{p1}_{h1} \\ & - 0.25 * (t\_vo^{p1}_{h2} * t\_vo^{p2}_{h1} * t\_vo^{p3}_{h3} * v\_oovv^{h2h3}_{p2p3}) \\ & + 0.25 * (t\_vvoo^{p2p1}_{h2h1} * t\_vo^{p3}_{h3} * v\_oovv^{h2h3}_{p2p3}) - 0.125 * (t\_vo^{p1}_{h2} * t\_vvoo^{p2p3}_{h3h1} * v\_oovv^{h3h2}_{p2p3}) \\ & - 0.125 * (t\_vo^{p2}_{h1} * t\_vvoo^{p3p1}_{h2h3} * v\_oovv^{h2h3}_{p3p2}) - 0.25 * (t\_vo^{p2}_{h1} * v\_ovvv^{h2p1}_{p2p3} * t\_vo^{p3}_{h2}) \\ & - 0.25 * (t\_vo^{p1}_{h2} * v\_oooov^{h2h3}_{h1p2} * t\_vo^{p2}_{h3}) - 0.25 * (t\_vo^{p1}_{h2} * t\_vo^{p2}_{h1} * f\_ov^{h2}_{p2}) \\ & + 0.125 * (t\_vvoo^{p2p3}_{h2h1} * v\_ovvv^{h2p1}_{p2p3}) + 0.125 * (t\_vvoo^{p2p1}_{h2h3} * v\_oooov^{h2h3}_{h1p2}) \end{aligned}$$

... Other computations that modify tensors t\_vo etc.

# Experiment: Optimized CCSD T1

$$it\_1^{q1h1}_{q2q3} = (a\_mmmm^{q4q1}_{q2q3} * c\_om^{h1}_{q4})$$

$$it\_2^{h1h2}_{p1q1} = (c\_mv^{q2}_{p1} * (c\_om^{h1}_{q3} * it\_1^{q3h2}_{q1q2}))$$

$$v\_oovv^{h1h2}_{p1p2} = (c\_mv^{q1}_{p1} * it\_2^{h2h1}_{p2q1})$$

$$it\_4^{h1h2}_{h3p1} = (c\_mo^{q1}_{h3} * it\_2^{h1h2}_{p1q1})$$

$$it\_3^{q1}_{h1} = (c\_mv^{q1}_{p1} * t\_vo^{p1}_{h1})$$

$$it\_5^{q1}_{q2} = (it\_1^{q1h1}_{q2q3} * it\_3^{q3}_{h1})$$

$$it\_6^{h1}_{p1} = (v\_oovv^{h1h2}_{p1p2} * t\_vo^{p2}_{h2})$$

$$\begin{aligned} \text{residual}^{p2}_{h1} = & 0.25 * f\_vo^{p2}_{h1} - 0.25 * (f\_oo^{h2}_{h1} * t\_vo^{p1}_{h2}) + 0.25 * (f\_vv^{p1}_{p2} * t\_vo^{p2}_{h1}) \\ & + 0.125 * (c\_vm^{p1}_{q1} * (it\_1^{q1h2}_{q2q3} * (c\_mv^{q1}_{p1} * (c\_mv^{q1}_{p2} * t\_vovo^{p2p1}_{h1h2})))) \\ & - 0.25 * ((f\_ov^{h1}_{p1} * t\_vo^{p1}_{h2}) * t\_vo^{p1}_{h2}) - 0.125 * ((t\_vo^{p2}_{h3} * v\_oovv^{h1h2}_{p1p2}) * t\_vovo^{p2p1}_{h2h3}) \\ & - 0.125 * ((t\_vovo^{p1p2}_{h3h2} * v\_oovv^{h3h1}_{p1p2}) * t\_vo^{p1}_{h2}) + 0.25 * (t\_vovo^{p2p1}_{h2h1} * it\_6^{h2}_{p2}) \\ & - 0.25 * ((it\_6^{h1}_{p1} * t\_vo^{p1}_{h2}) * t\_vo^{p1}_{h2}) - 0.25 * (c\_vm^{p1}_{q1} * (c\_mo^{q2}_{h1} * it\_5^{q1}_{q2})) \\ & - 0.25 * (c\_vm^{p1}_{q1} * (it\_3^{q2}_{h1} * it\_5^{q1}_{q2})) + 0.125 * (it\_4^{h2h3}_{h1p2} * t\_vovo^{p2p1}_{h3h2}) \\ & - 0.25 * ((it\_4^{h3h1}_{h2p1} * t\_vo^{p1}_{h3}) * t\_vo^{p1}_{h2}) + 0.25 * (t\_vovo^{p2p1}_{h2h1} * f\_ov^{h2}_{p2}) \end{aligned}$$

... Other computations that modify tensors t\_vo etc.

# Experimental Results: CCSD T1

Optimizing **CCSD T1** ( $O=10$ ,  $V=500$ )

Iteration Count	Expanded MO Tensors	Operation Count	Reduction Factor
1	-	$5.36 \times 10^{12}$	1
	v_ovvv	$1.59 \times 10^{12}$	3.37
	v_ovvv, v_ooov, v_ovov	$1.51 \times 10^{12}$	3.55
10	-	$5.63 \times 10^{12}$	1
	v_ovvv	$2.34 \times 10^{12}$	2.41
	v_ovvv, v_ooov, v_ovov	$2.26 \times 10^{12}$	2.49

# Experimental Results: CCSD T2

## Optimizing **CCSD T2**

Iteration Count	Expanded MO Tensors	Operation Count	Reduction Factor
1	-	$2.85 \times 10^{14}$	1
	v_vvvv	$2.72 \times 10^{13}$	10.48
	v_vvvv, v_ovvv, v_vvov	$1.93 \times 10^{13}$	14.75
10	-	$4.22 \times 10^{14}$	1
	v_vvvv	$1.76 \times 10^{14}$	2.40
	v_vvvv, v_ovvv, v_vvov	$1.67 \times 10^{14}$	2.53

# Kernel Optimization: Matrix Transpose

- A naïve implementation of Matrix Transposition :

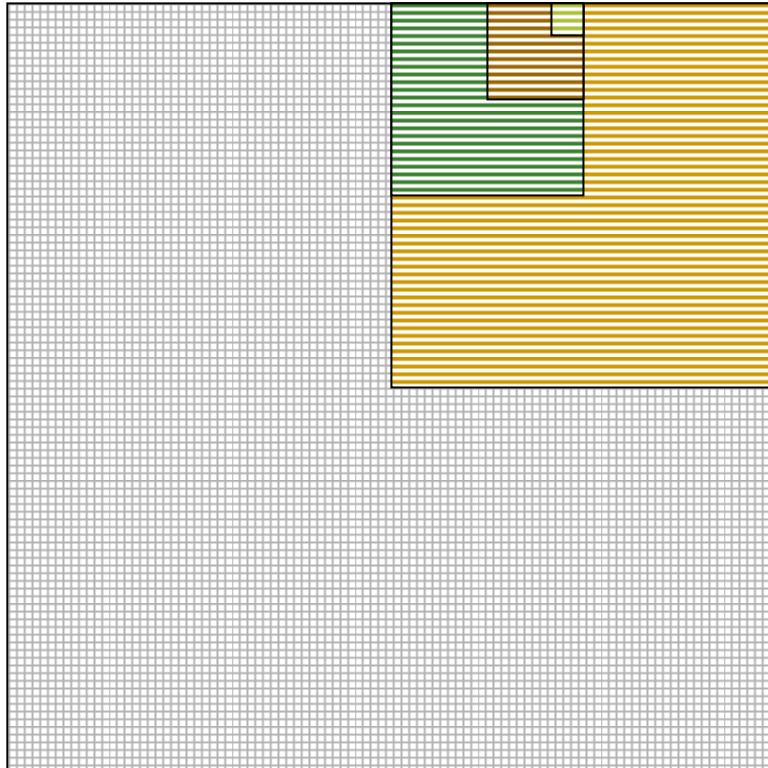
```
for (i = 0; i < N1; i++)  
    for (j = 0; j < N2; j++)  
        B[i][j] = A[j][i];
```

- On a Pentium 4 with gcc 8.0:
  - Average bandwidth achieved: 90.3 MB/s
  - Best copy bandwidth reported by the STREAM benchmark: 3.2GB/s (**35.5X**)
- Need to better exploit spatial locality (L1, L2, TLB)
- Architecture-specific optimizations are needed
  - SIMD support (SSE)
  - Consideration on memory subsystems
  - ...

# Optimization Issues

Cache hierarchy	Spatial locality; Minimizing conflict misses; Reducing TLB misses.
Memory subsystem	Improving memory bandwidth utilization; Reducing memory-bus turnaround overhead; Exploiting efficient DRAM accesses.
SIMD instructions	Exploiting register-level data reorganization; Accessing data with alignment constraints; Exploiting cacheability control instructions.

# Four Levels of Tiling



  
Register  
Level

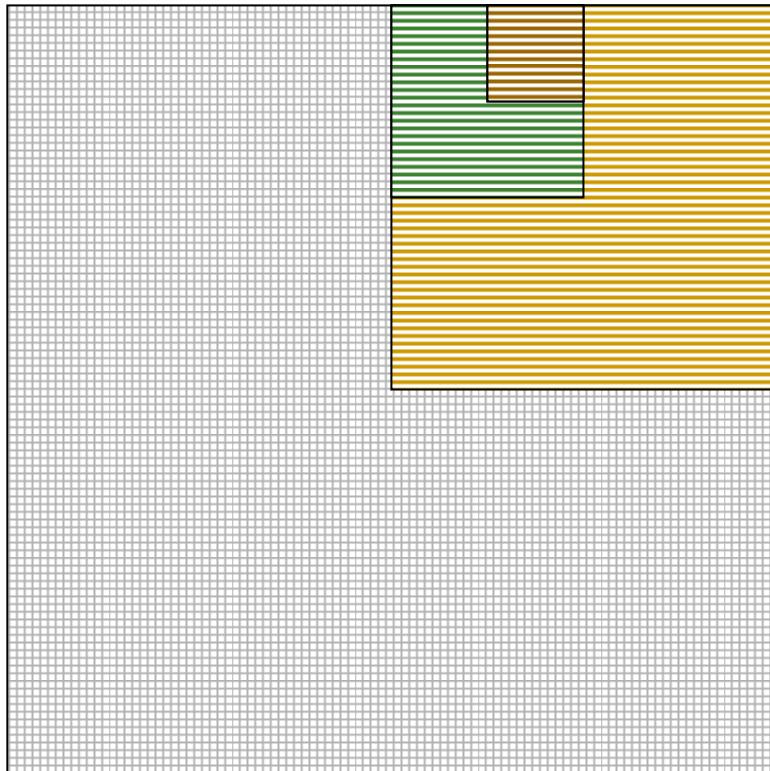
  
Cache  
Level 1

  
Cache  
Level 2

  
TLB  
Level

# Using a Buffer

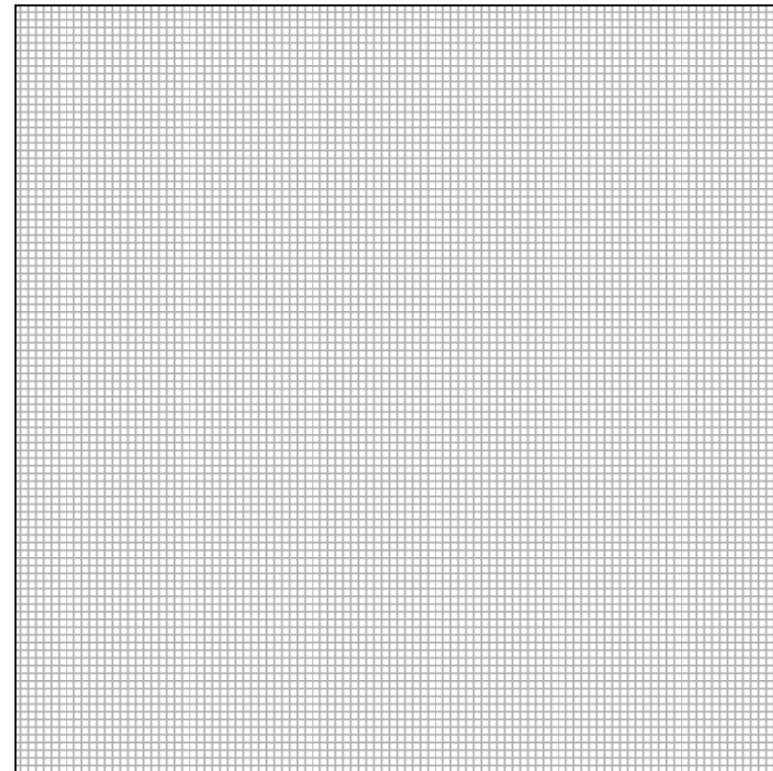
## Buffering off Reads



  
Cache  
Level 1

  
Cache  
Level 2

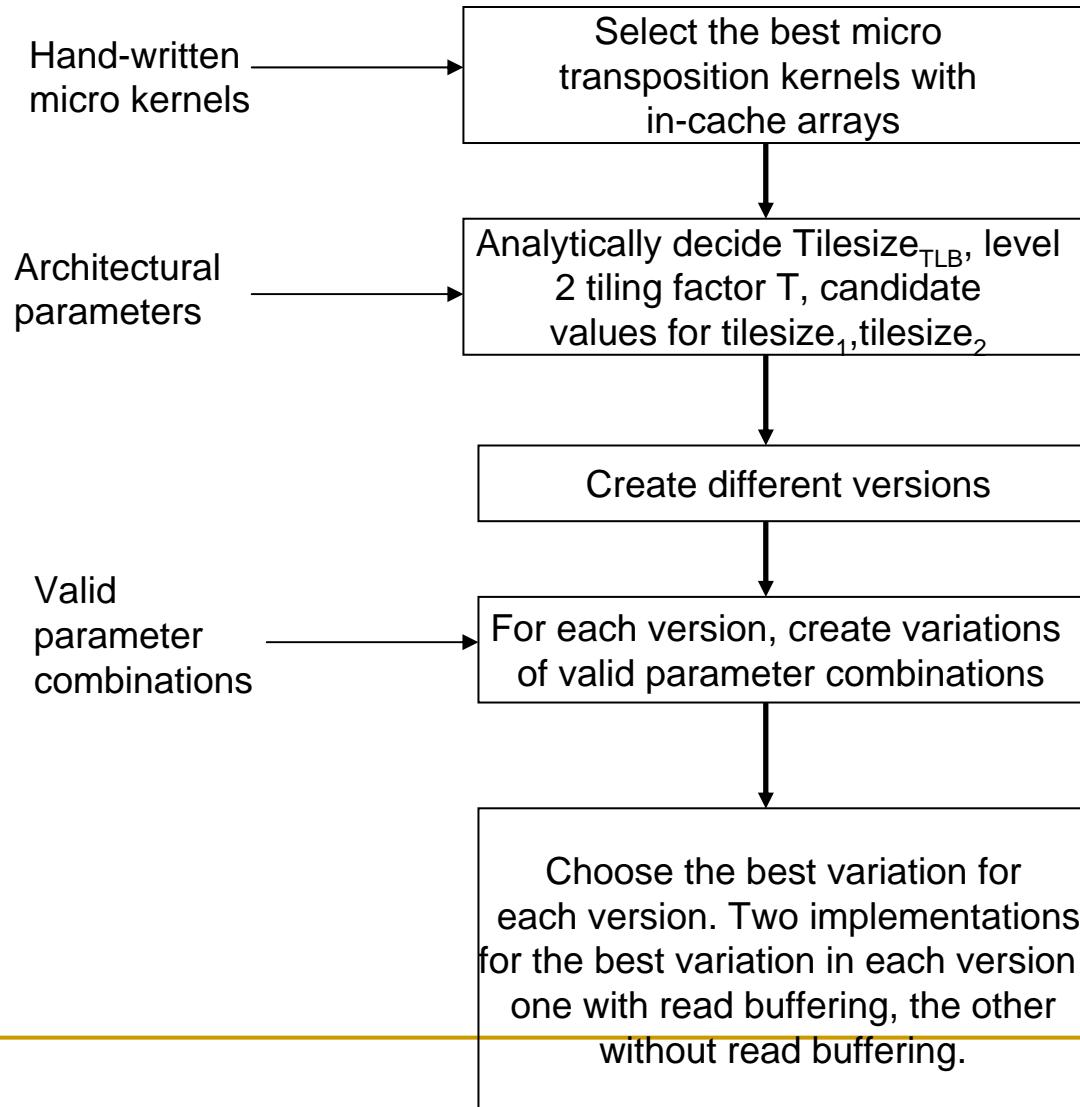
  
TLB  
Level



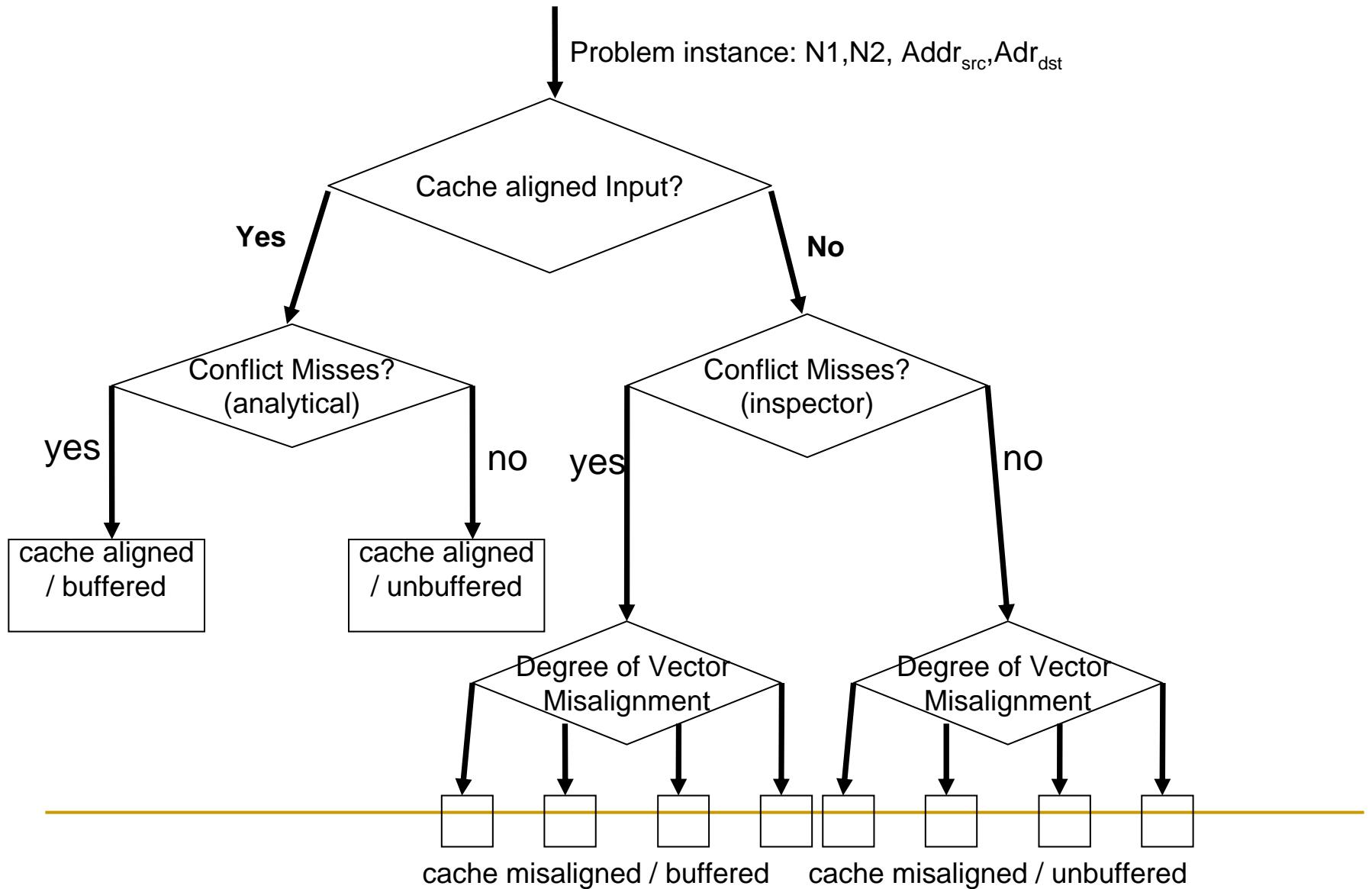
# Optimization Parameters

- Register level tiling (micro-transposition)
  - The best SIMD kernel for each alignment: **empirical**
  - Inter-tile loop order (i,j) or (j,i): **empirical**
- Cache level 1 tiling (mini-transposition)
  - Tile size ( $\text{tilesize}_1, \text{tilesize}_2$ ): **empirical**;
  - Inter-tile loop order ( $iT, jT$ ) or ( $jT, iT$ ): **empirical**
  - Whether to have buffer copies: **analytical + empirical**
- Cache level 2 tiling:
  - Tile size  $T^*L_{2e}$ : **analytical**
- TLB level tiling:
  - Tile size  $\text{Tilesize}_{\text{TLB}}$ : **analytical**

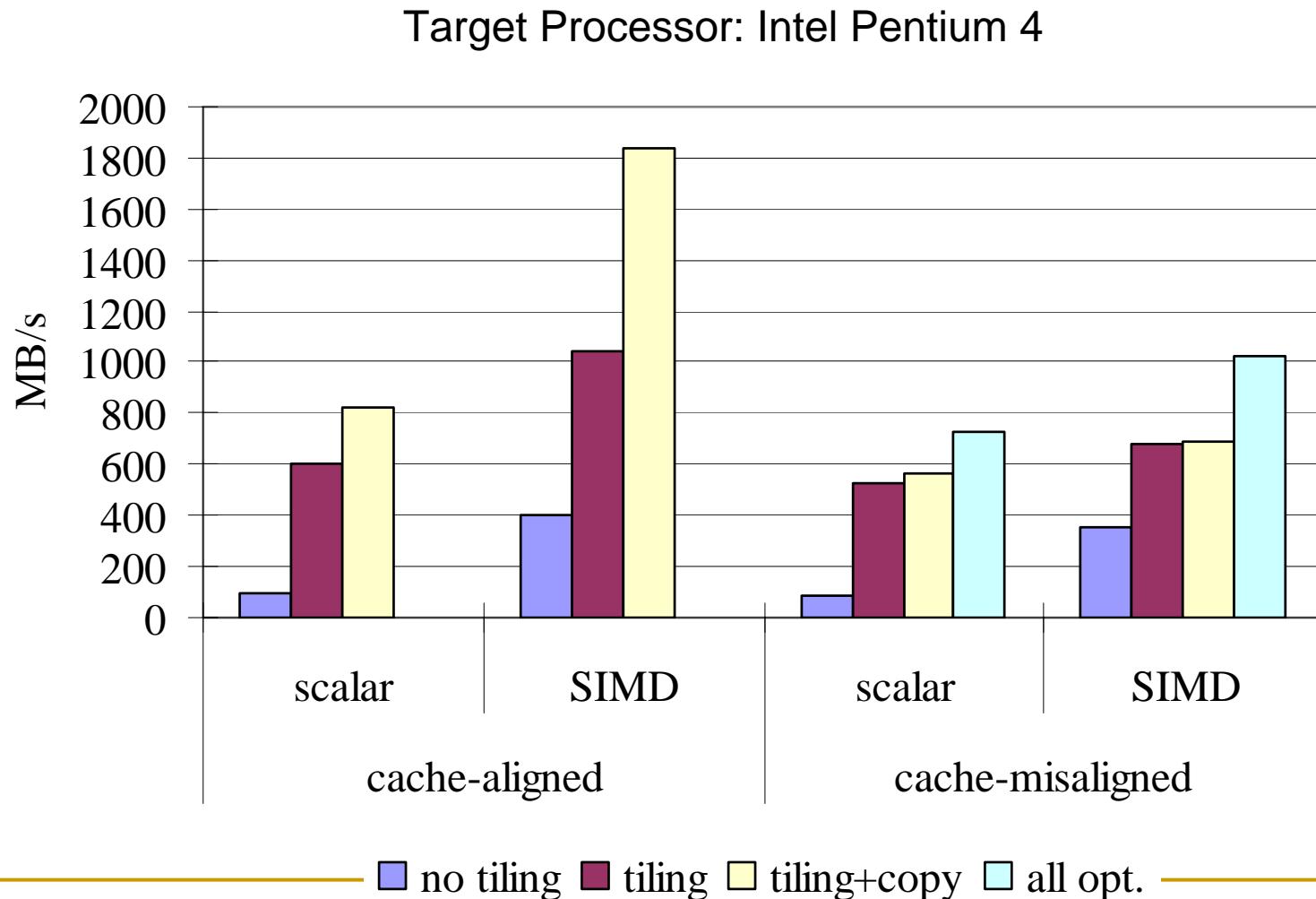
# Library Code Generation Procedure



# Runtime Decision Tree



# The Impact of Optimizations



# Petascale SW Development Challenges

- Currently available programming environments are inadequate
  - Too much effort required to achieve high performance on complex applications
  - Available abstractions for data and control are inadequate, especially for dynamic, irregularly structured applications

Data Model	Computation Model			
	Implicitly Parallel	Computation-centric		Process-centric Parallel
		Iterators	Function-centric	
Uniform Global	HPF, PETSc, Collectives	OpenMP	Cilk, X10, Chapel, OpenMP	UPC, GA, OpenMP
Partitioned	SCALAPACK	X10	X10	MPI, UPC, GA, CAF

# Need for Multiple Interoperable Views

- Different models have benefits w.r.t. productivity/ performance
- Need multiple inter-operable views of data and control
- Use high-level abstract view (high-productivity) for non-performance critical parts of code, and lower-level concrete view only for performance-critical regions

# Blocked-Global Data View

Data Model	Computation Model			
	Implicit	Computation-centric		Process-centric
		Iterators	Function-centric	
Uniform Global	HPF, PETSc, Collectives, <b>BSTL</b>	OpenMP	Cilk, X10, OpenMP	UPC, GA, OpenMP, <b>BSTL</b>
Blocked Global		<b>BSTL</b>		<b>BSTL</b>
Partitioned	SCALAPACK	X10	X10	MPI, UPC, GA, CAF, <b>BSTL</b>

- Blocked global view organizes shared data in memory-contiguous chunks distributed over the systems memory-spaces, enabling efficient movement and locality-aware load-balancing by system

# The Data-Movement Complexity of Algorithms

- The currently used primary FLOPs metric will become increasingly irrelevant
  - Operations are cheap, data movement is expensive!
  - Off-chip data bandwidth will become primary bottleneck for massively multi-core processors
- The maximum performance achievable with these two algorithms is very different; need model to characterize data-movement complexity

```
f = 1
for k = 1 to N
    s = 0
    for i = 1 to N
        for j = 1 to N
            C[i,j] += k*A[i,k]*B[k,j]
            s += C[i,j]
        end
    end
f = f + s/(N*N)
end
```

```
f = 1
for k = 1 to N/
    s = 0
    for i = 1 to N
        for j = 1 to N
            C[i,j] += f*A[i,k]*B[k,j]
            s += C[i,j]
        end
    end
f = f + s/(N*N)
end
```

# Conclusions

- Scalable petascale software development will be very challenging without advances in systems software: domain-specific and general-purpose
- Need teams of systems software developers and application experts to make needed advances
  - HPCS languages address some issues; but development of diverse set of significant applications must be a high priority
- Interesting algorithm development challenges await us as we go massively multi-core