Performance benchmark results for ACES III

Erik Deumens, Victor Lotrich, Mark Ponton, Tomasz Kus, Norbert Flocke, Anthony Yau, Rod Bartlett

AcesQC, LLC

Gainesville, Florida

DoD User Group Conference 2008, Seattle

Outline of the talk

- Challenges in computational chemistry
  - Larger, more complex molecules
- Results for molecules of interest
  - What can be done today?
- Challenges in high performance computing
  - Parallelism at many levels

Computational Chemistry Challenges

- Larger molecules
  - Nano structures
  - Molecular electronics
  - Polymer properties
  - Biological molecules
- More emphasis on processes
  - Reaction description rather than transition probabilities

Computational challenges

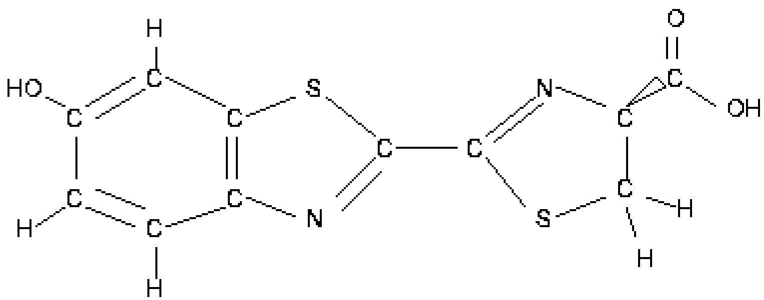
- Energies
- Gradients
- Geometry search
  - Equilibrium configuration
  - Transition states
- Vibrational frequencies
- Excited electronic states

Outline of the talk

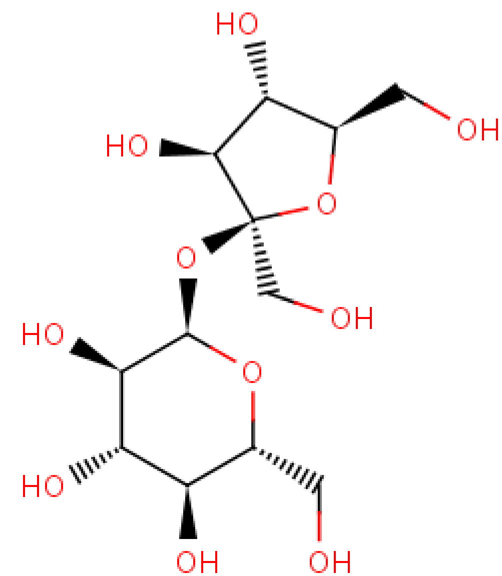
- Challenges in computational chemistry
  - Larger, more complex molecules
- Results for molecules of interest
  - What can be done today?
- Challenges in high performance computing
  - Parallelism at many levels

CCSD(T)

- Luciferin($C_{11}H_8O_3S_2N_2$)
- RHF
- C_1 symmetry
- Basis = aug-cc-pvdz (494 bf)
- $N^{corr}_{occc} = 46$



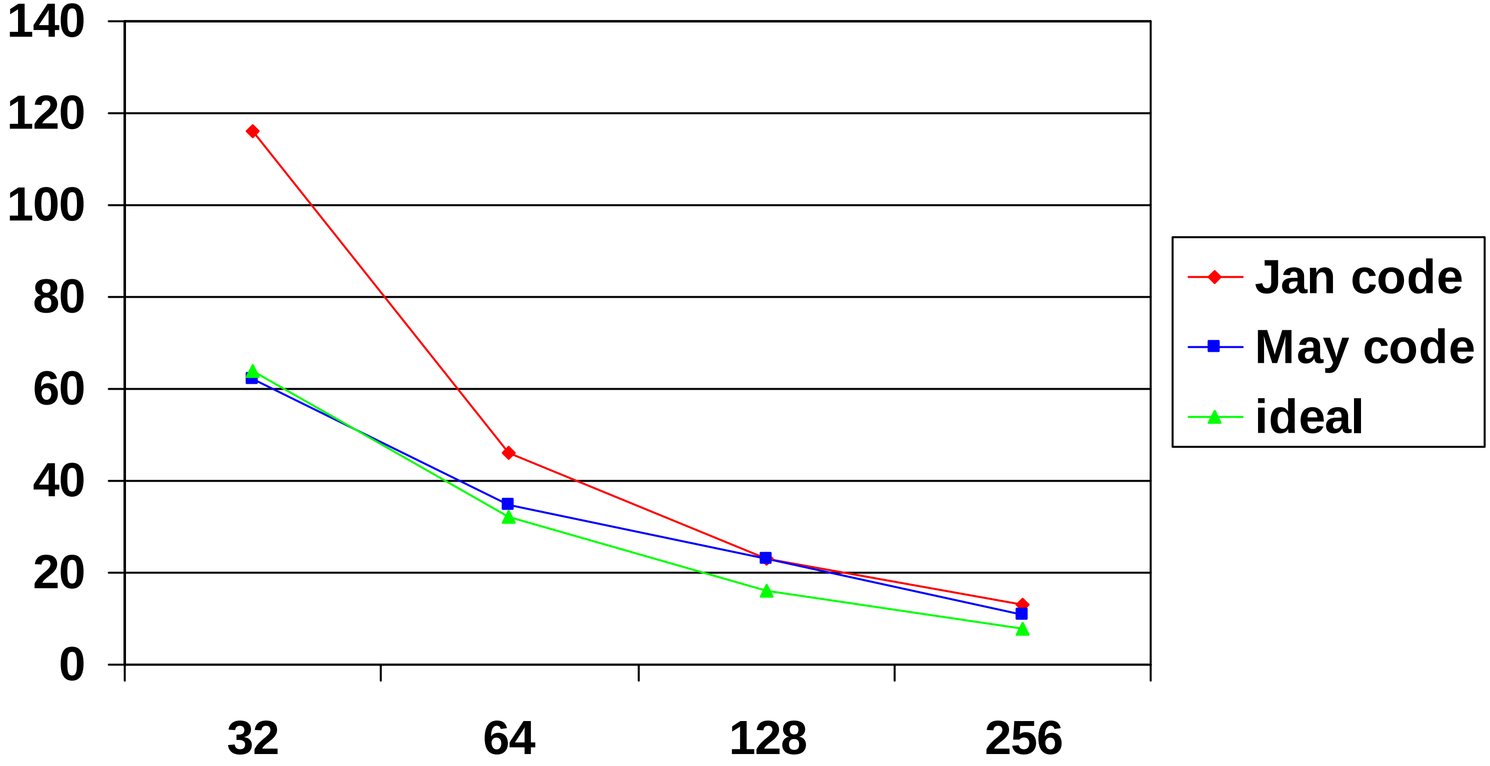
- Sucrose ($C_{12}H_{22}O_{11}$)
- RHF
- C_1 symmetry
- Basis = 6-311G** (546 bf)
- =68



ACES III software

- Developed under CHSSI CBD-03
- Parallel for shared and distributed memory
- Capabilities
  - Hartree-Fock (RHF, UHF)
  - MBPT(2) energy, gradient, hessian
  - CCSD(T) energy and gradient (DROPMO)
  - EOM-CC excited state energies

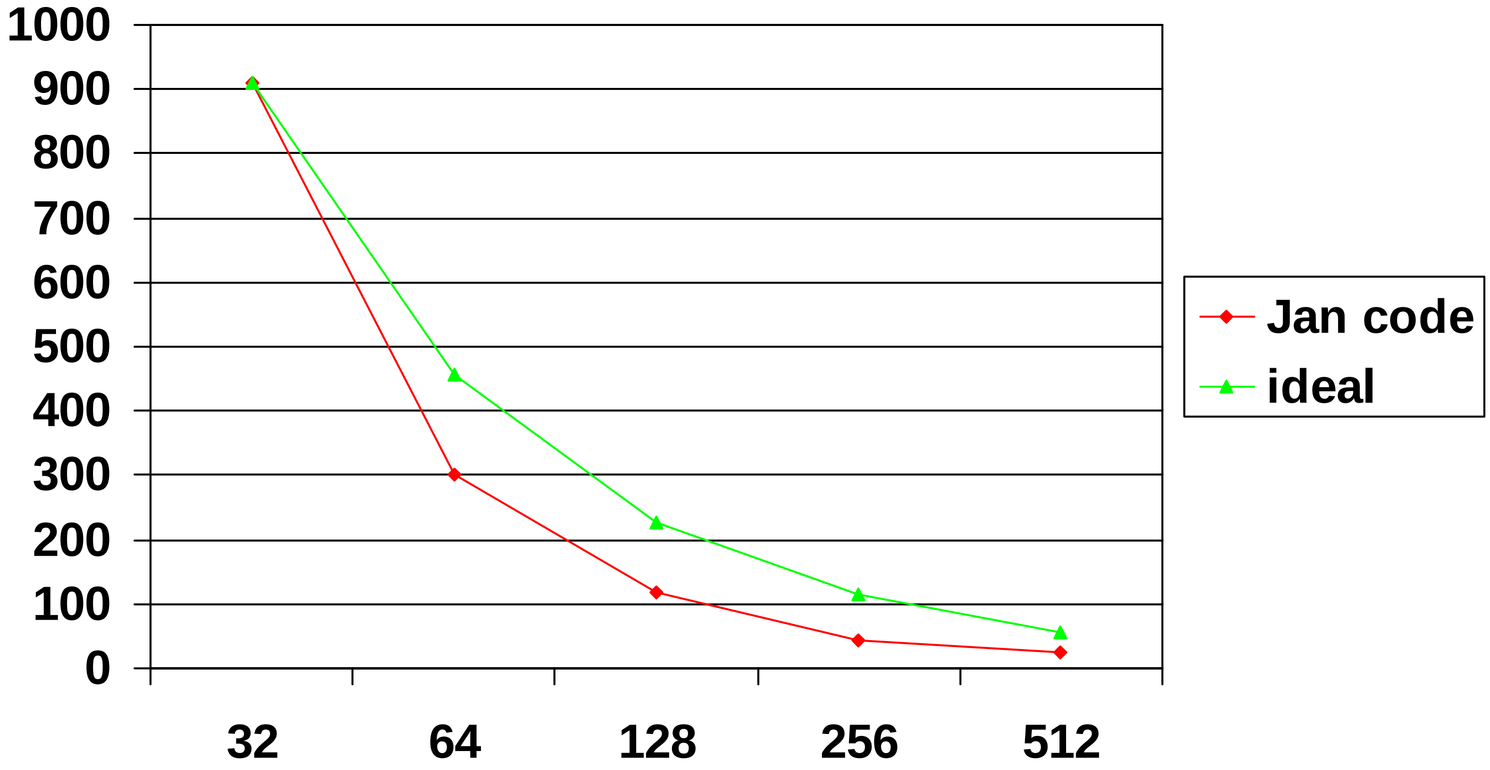
Luciferin CCSD scaling
min per iter; 12 iterations; two versions;



Luciferin CCSD(T)

- CCSD on 128 processors
  - One iteration: 23 min
  - Total 12 iterations: 275 min
- (T)
  - Hardest 8 occupied orbitals: 420 min on 128 processors
  - Total 48 correlated orbitals: 420 min on 768 processors

Sucrose CCSD scaling
min per iteration



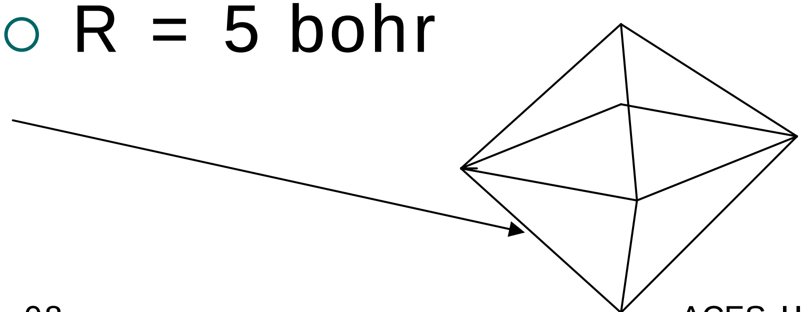
Succrose CCSD super linear scaling

- CCSD iteration
  - 32 processors 909 min
  - 512 processors 24 min, ideal: 57 min

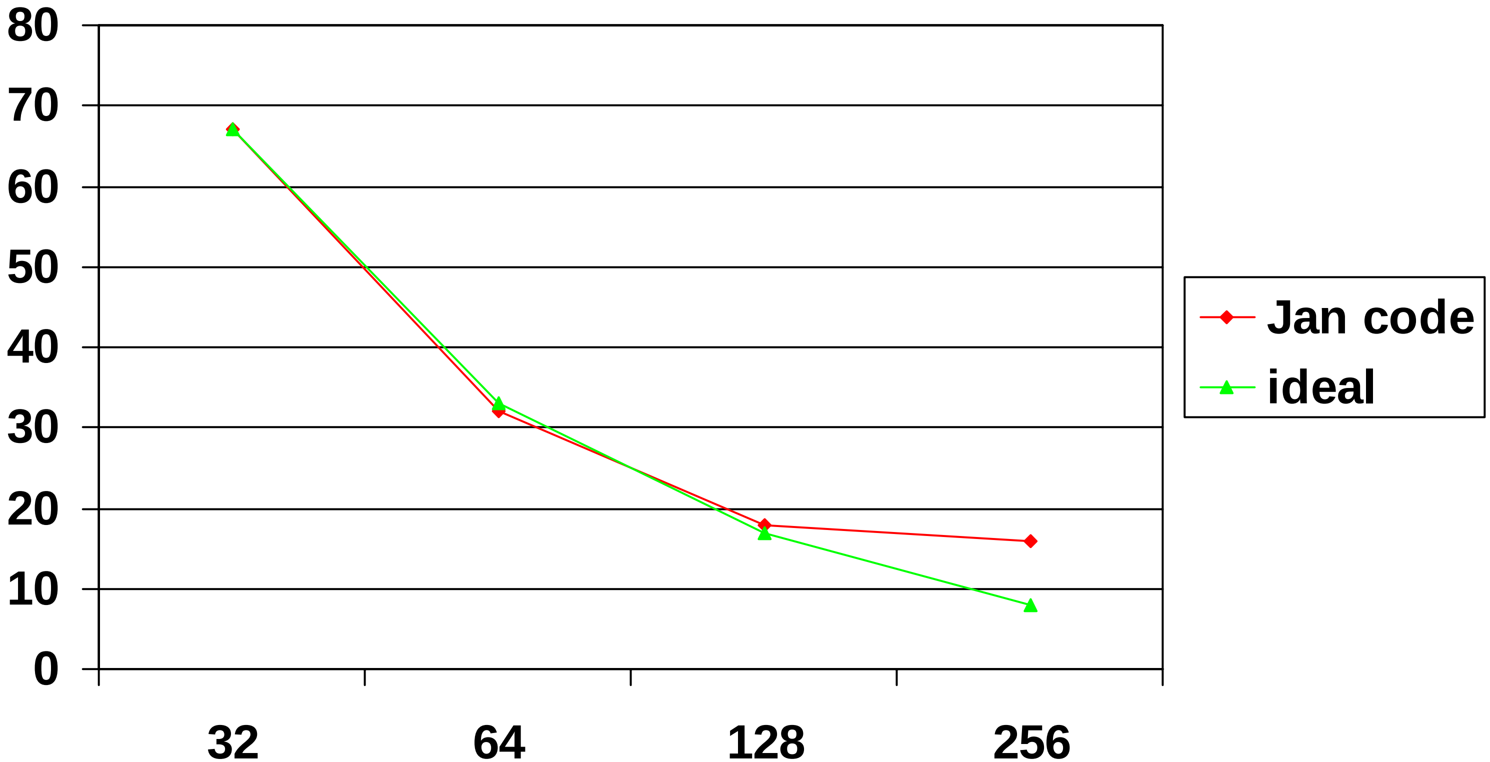
Ar_N Cluster Benchmarks(Performance)

- <u>Specifications</u>
- N=6
- UHF
- C_1 symmetry
- Basis = aug-cc-pvtz (300bf)
- N^{corr}_{occ} = 54
- R = 5 bohr

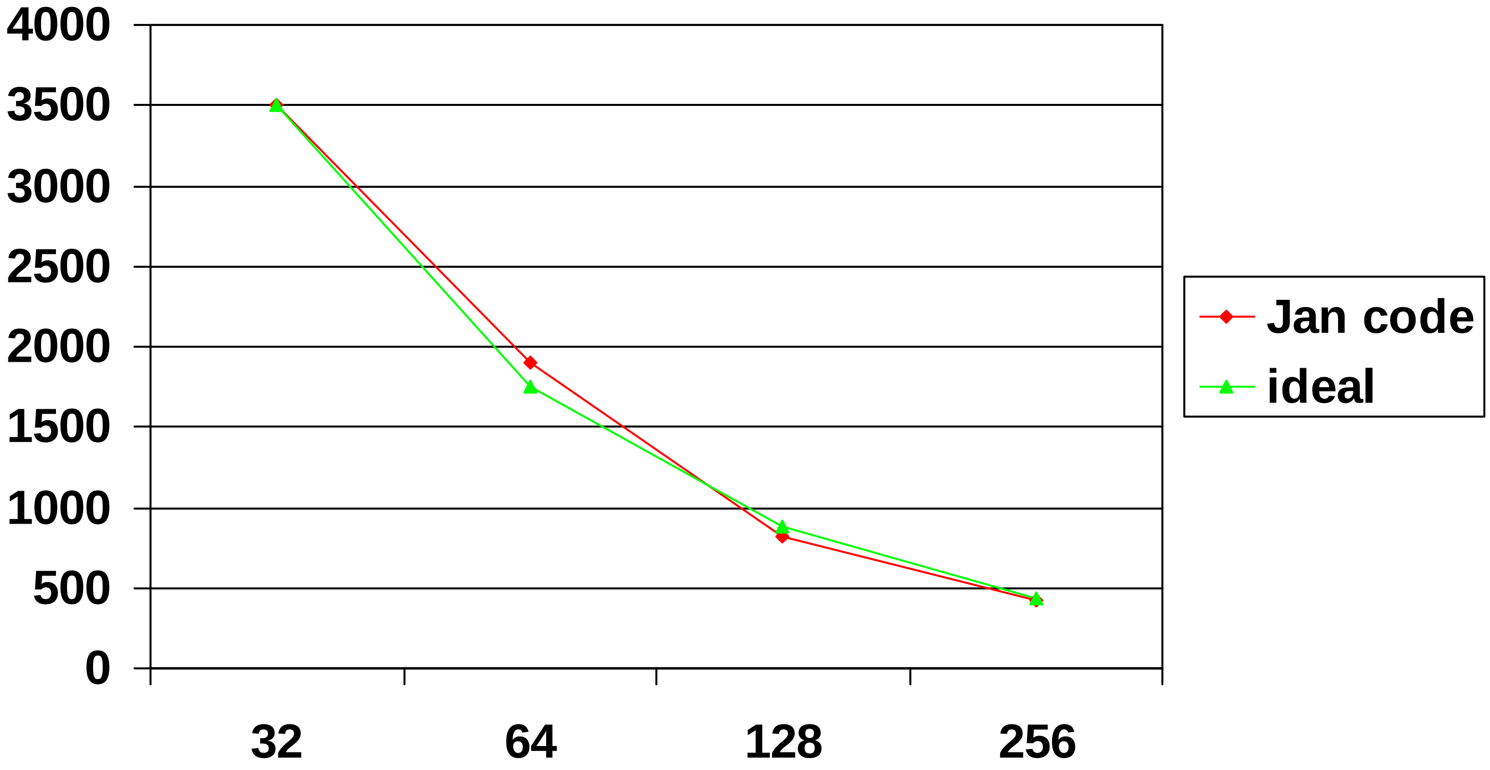
- <u>Methods</u>
- MBPT(2) gradient
- CCSD gradient
- CCSD(T) (core dropped)
- MBPT(2) Hessian (RHF)



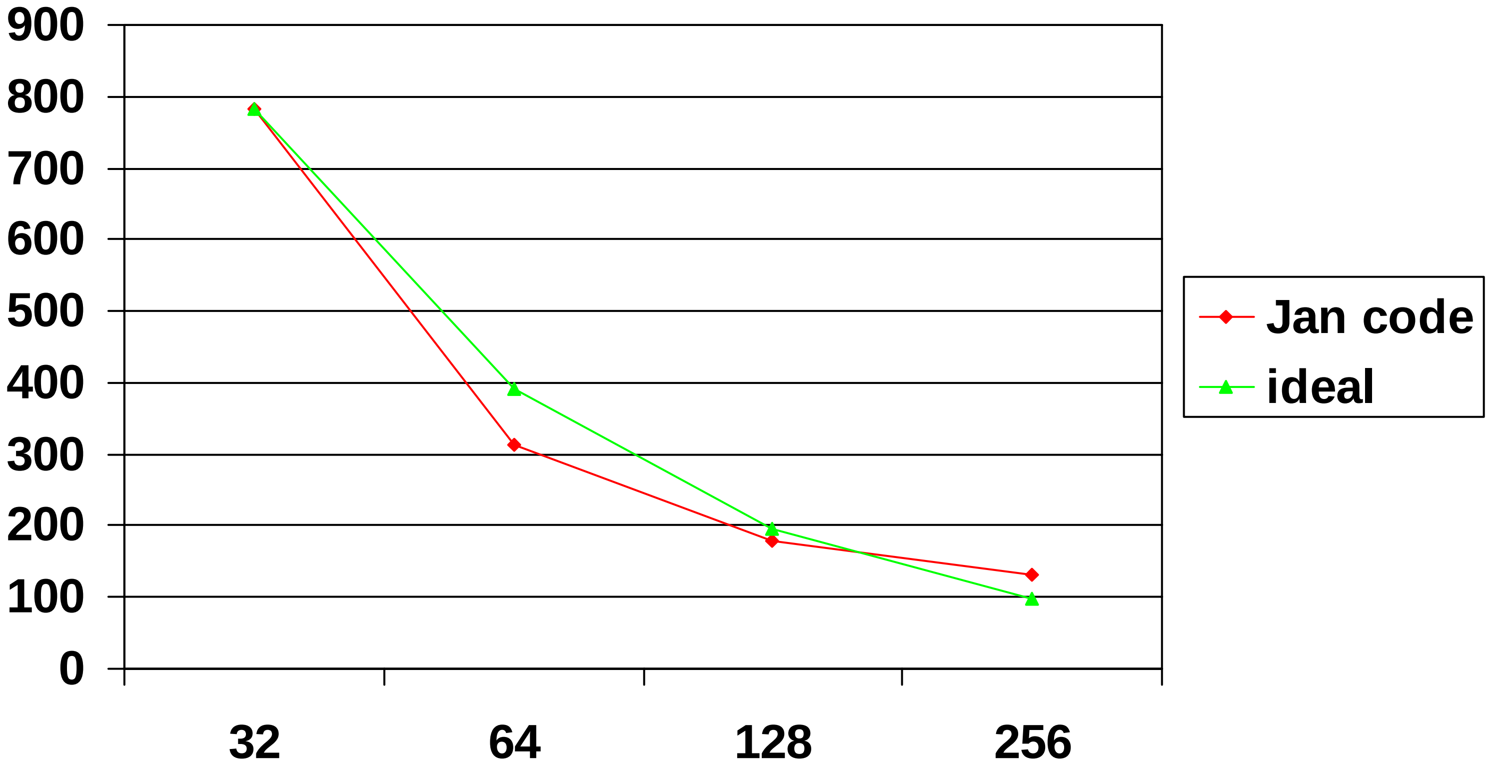
Ar_6 UHF MBPT(2) gradient scaling min per iteration; 54 corr occ alpha



Ar_6 UHF CCSD gradient scaling min per iteration; 54 corr occ alpha



Ar_6 UHF CCSD(T) scaling
min per iteration; 24 corr occ alpha



Ar_6 MBPT(2) Hessian Results

- Asymmetric evaluation algorithm
- $V^* d^2V/dpdq$
- dV/dp
- $d\mathcal{V}/dq$
- $dV/dp^* d\mathcal{V}/dq$
- Number of Hessian elements = 324/2

- Number of processors = 128
- T=381 minutes
- 155 sec / pert p
- 330 sec / pert q
- 16 sec / element

Benchmarks website

- From workshop on “Parallelization of Coupled Cluster Methods”
  - Feb 23-24, 2008 St. Simons Island, Georgia at 2008 Sanibel Symposium
- http://www.qtp.ufl.edu/PCCworkshop/PCCbenchmarks.html

Outline of the talk

- Challenges in computational chemistry
  - Larger, more complex molecules
- Results for molecules of interest
  - What can be done today?
- Challenges in high performance computing
  - Parallelism at many levels

A single CPU computer

- Basic data item: 64 bit number
- High level language: Fortran, C
  - c = a + b
- Assembly language
  - ADD dest,src
  - ADD is an operation code
  - dest and src are registers

The ACES III machine

- Basic data item: data block 10,000 64 bit numbers -> super number
- High level language: being developed
- Assembly language: SIAL super instruction assembly language
  - R(I,J,K,L) += V(I,J,C,D) * T(C,D,K,L)
- xaces3 -> super instruction processor

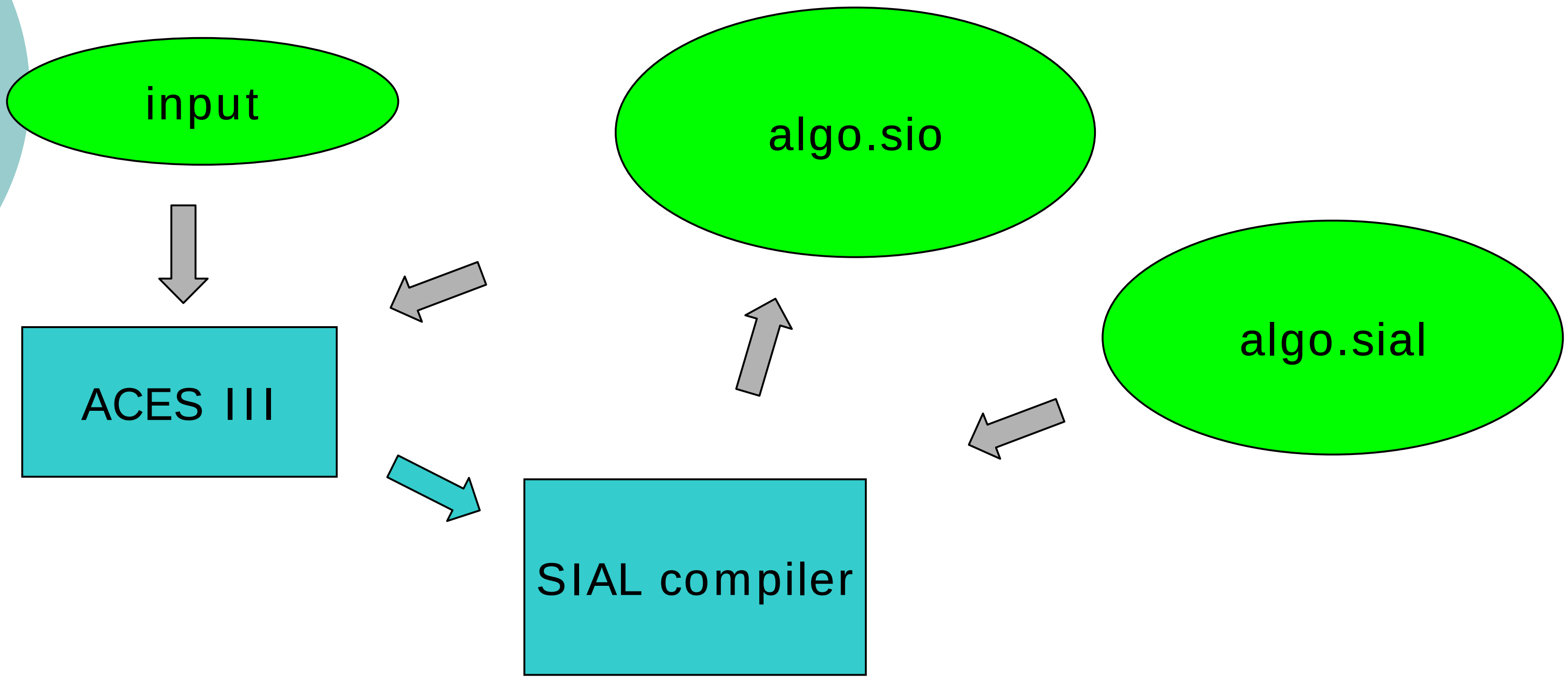
Coarse grain parallelism

- Memory super instruction
  - GET block
- can be from
  - Local node RAM
  - Other node RAM
- Only difference is execution time!

Fine grain parallelism

- Compute super instruction
  - * (contractions)
  - compute_integrals
- can use multiple cores and accelerators like GPU

Execution flow



Distributed data flow

- N worker tasks each with local RAM
- Data distributed in RAM of workers
  - AO-based: direct use of integrals
  - MO-based: use transformed integrals
- Array blocks are spread over all workers
- Workers compute integrals when integral instruction is called

Disk resident data flow

- M server tasks
  - have access to local or global disk storage
  - accept, store and retrieve blocks
  - also compute integrals when asked
- Data served to and from disk

New developments

- Develop higher level programming language
- Data staging
  - Huge served array
  - Copy section in distributed array
  - Work efficiently on distributed array
- Similar to BLAS-3 management of cache

ACES III is ready

- Tackle problems larger than ever
- We are now working on getting some benchmark results on 1,000 and 2,000 processors
  - The problem is getting quick access to 1,000 and 2,000 processors to tune for good performance
  - Running ACES III is easy