Gaussian Tutorial: Estimating Resource Requirements

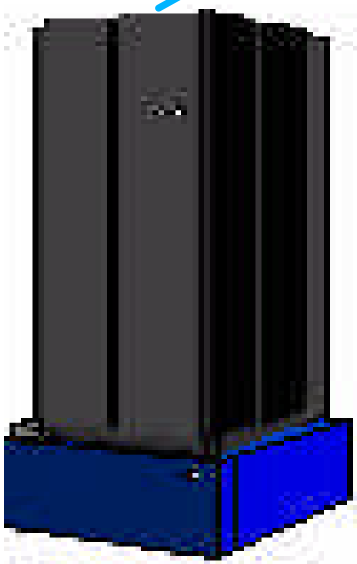
Carlos P. Sosa
IBM
and
Patton Fast
Supercomputing Institute

Contents

- IBM SP Overview
- Review common methods
- Alternative algorithms and why the program selects them
- Estimating resource usage
- Allocating memory and disk resources for good performance
- Running Gaussian

IBM SP Overview

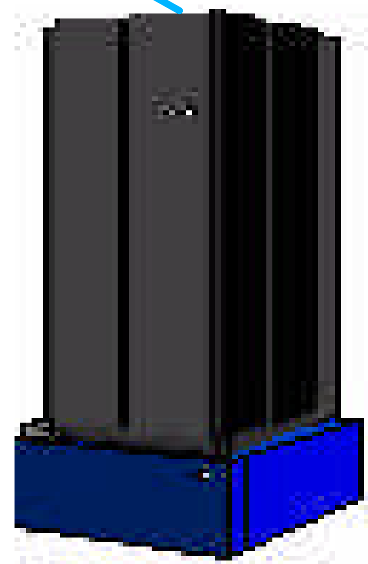
UNIVERSITY OF MINNESOTA
Supercomputing Institute
for Digital Simulation and Advanced Computation
1200 Washington Ave S - Minneapolis, MN 55415
Phone: (612) 625-1818 - Fax: (612) 624-8861



78 WinterHawkII, 375 MHz

17 NightHawk, 222 MHz

4 Silver nodes, 333 MHz



Nodes are 4-way

Silver nodes are 333 MHz, 604e, 32-bit

Gaussian Design

- *Gaussian98* has been designed to work efficiently given a variety of computer configurations
- The program attempts to select the most efficient algorithm given the memory and disk constraints imposed upon it
- *Gaussian98* offers a wide variety of algorithms
- An understanding of the possibilities and tradeoffs can help you to achieve optimal performance

Gaussian Input

```
%chk=h2o
%nproc=1
%mem=8MW
#p hf/sto-3g opt

water optimization

0 1
o
h 1 oh
h 1 oh 2 a

oh = 0.89
a = 105.
```

Control files & system resources

Choice of computational model

Type of calculation

Charge and multiplicity

Coordinates

Link 0 Commands

%mem=N Sets the amount of dynamic memory used to N works (8N bytes). The default is 6MW. N may be followed by a units designation: KB, MB, GB, KW, MW or GW

%nproc(l)=N Requests that the job use up to N processors

%chk=file Locates and names the checkpoint file

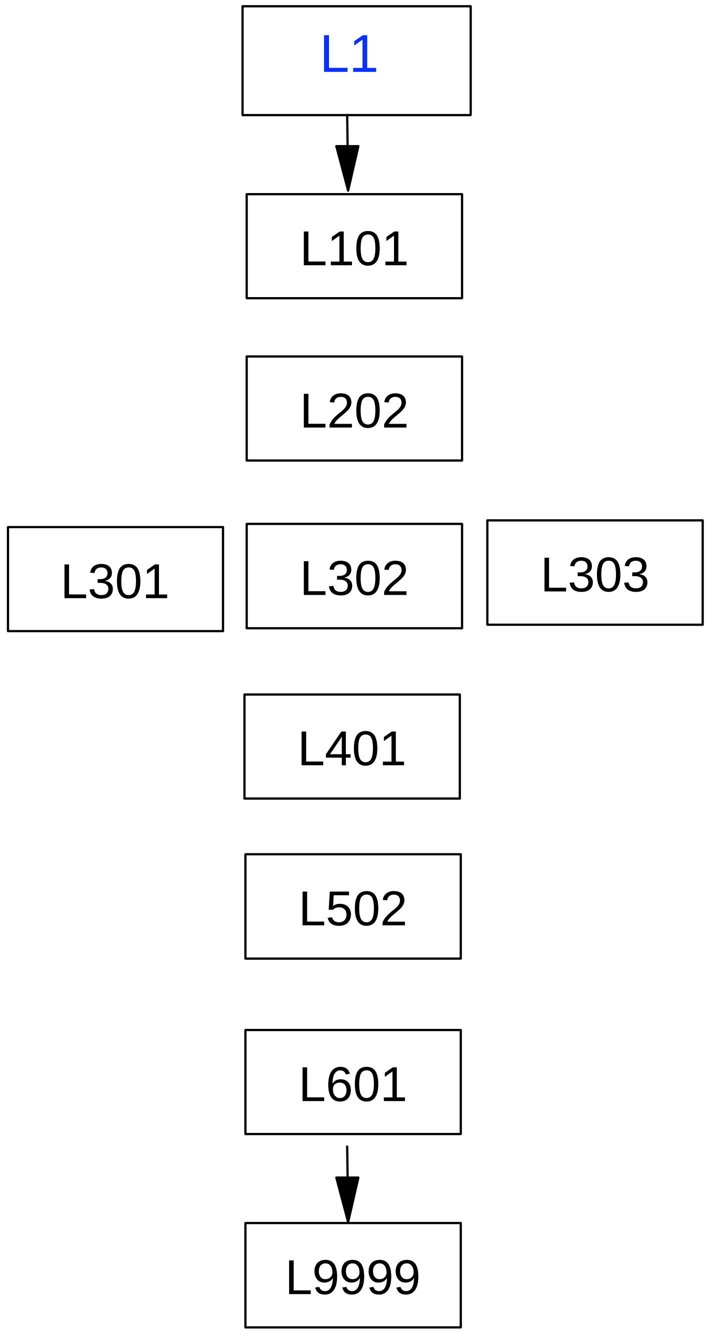
%rwf=file Locates and names a single, unified Read-Write file

%KJob LN [M] Tells the program to stop the run after the M^{th} occurrence of link N

%save Causes Link 0 to save scratch files at the end of the run

%subst LN dir Tells Link 0 to the executable for a link from alternate directory

Sequence of Programs



Read and parse route section

Read in molecule specification

Determine molecular symmetry

Set up basis set, compute one-electron integrals

Generate initials orbitals

Solve SCF equations

Assign orbital and wavefunction symmetries, print orbitals, and perform Mulliken population analysis

Hartree-Fock Energies

$$E_{HF} = \frac{\langle \Psi_o | H | \Psi_o \rangle}{\langle \Psi_o \Psi_o \rangle}; \qquad \frac{\partial E_{HF}}{\partial C_{\mu i}} = 0$$

$$\sum_{\nu} F_{\mu\nu} C_{\nu i} = \varepsilon_i \sum_{\nu} S_{\mu\nu} C_{\nu i}$$

$$F_{\mu\nu} = h_{\mu\nu} + \sum_{\lambda\sigma} [(\mu\nu \parallel \lambda\sigma) - (\mu\sigma \parallel \lambda\nu)] P_{\lambda\sigma}$$

$$P_{\lambda\sigma} = \sum_{i} C^{*}_{\lambda i} C_{\sigma i}$$

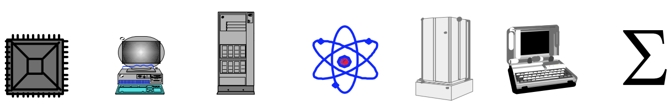
$$E_{HF} = \sum_{\mu\nu} P_{\mu\nu} h_{\mu\nu} + \frac{1}{2} \sum_{\mu\nu\lambda\sigma} \left[P^{T}_{\mu\nu} P^{T}_{\lambda\sigma} - P^{a}_{\mu\sigma} P^{a}_{\lambda\nu} \right] (\mu\nu \parallel \lambda\sigma) + V_{nuc}$$

Σ

Two-electron Integrals

Traditional approach:

- Formally $O(N^4)$; often less in practice
- Atomic Orbital (AO) basis:
  - Integrals in AO basis stored on disk in random order
  - Traditional approach for SCF
  - Sorting into standard order involves substantial extra storage
- Molecular Orbital (MO) basis:
  - Integrals transformed from AO to MO
  - Stored on disk in addition to AO integrals
  - Traditional approach beyond SCF



Two-electron Integrals

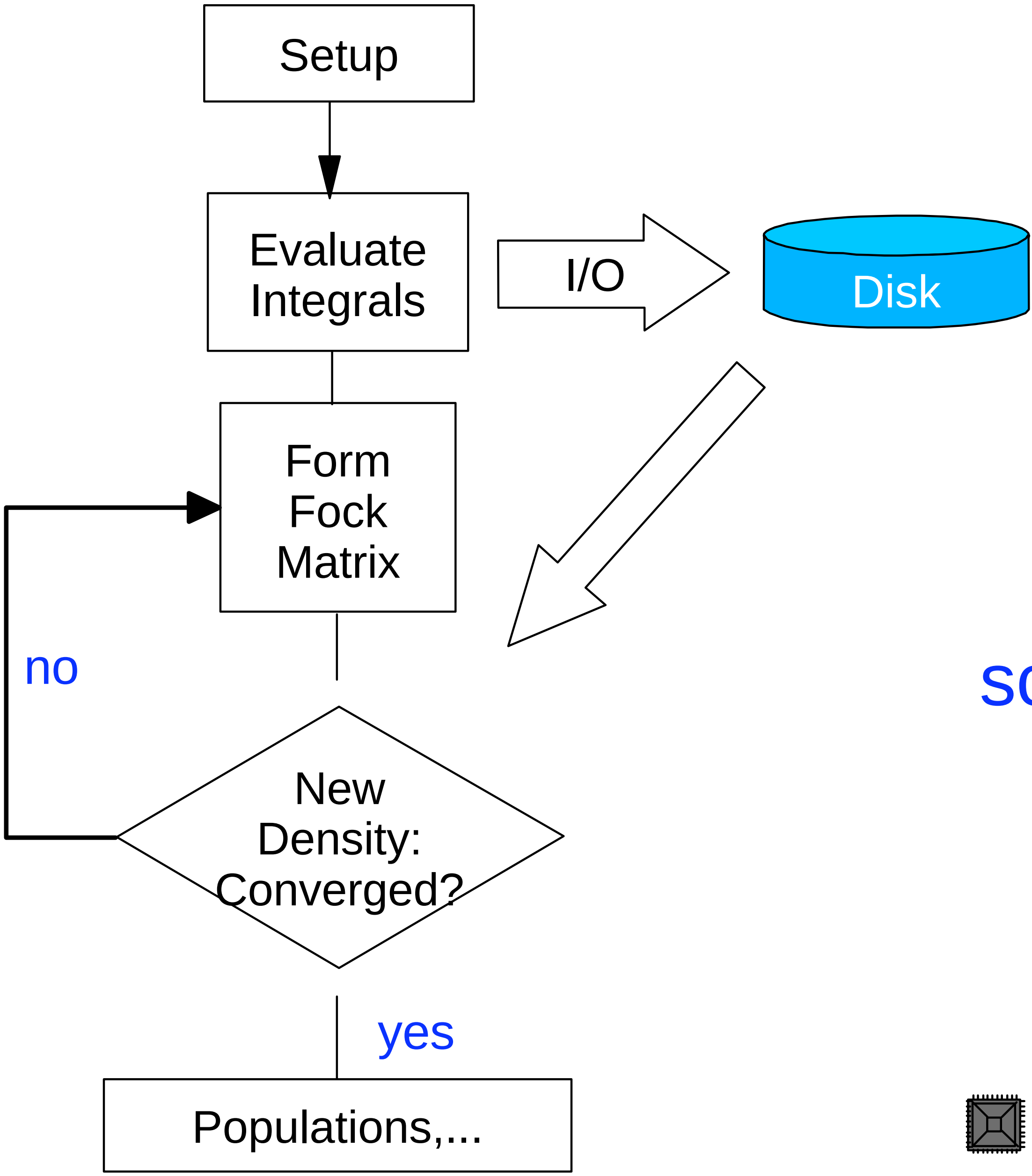
- Incore:
  - AO integrals are stored in main memory
  - Canonical order, including zeros
  - No I/O
  - Ordering facilitates optimization
- Direct:
  - Recompute integrals as needed

Direct SCF

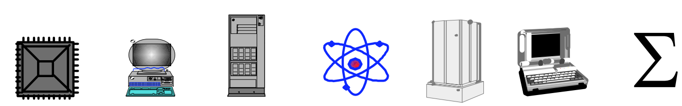
- Traditional approach:
  - *Integrals are expensive*
  - *Compute integrals once and store*
  - *Read integrals once each SCF iteration*
- Almlof:
  - *Integrals aren't that expensive*
  - *I/O can be slow*
  - *Amount of disk limits size of calculations*
  - *Recompute integrals each SCF iteration*
- Can be clever about neglecting integrals if their use is known:

$$\Delta F^{(n)} = F^{(n)} - F^{(n-1)} = \sum \Delta P^{(n)}_{\lambda\sigma} \langle \mu\lambda \| \nu\sigma \rangle$$

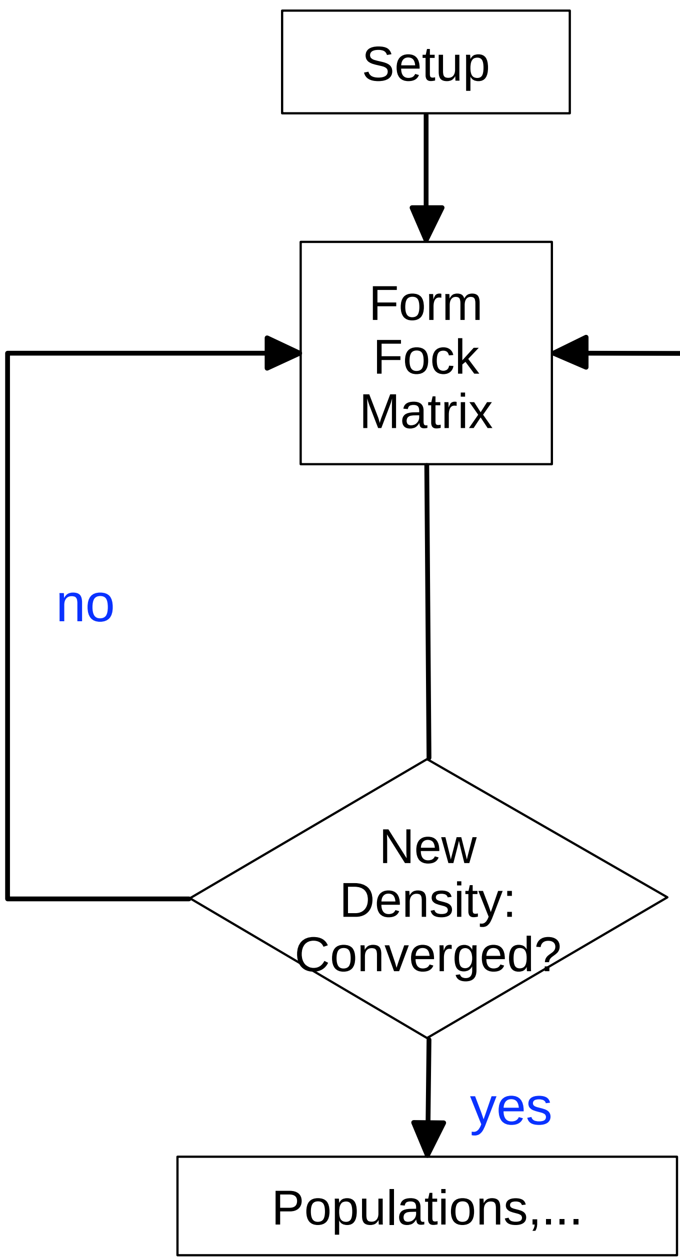
Conventional SCF



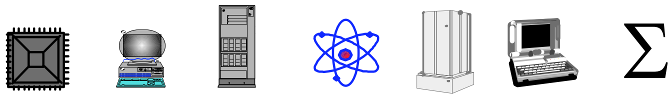
scf=conventional



Direct SCF

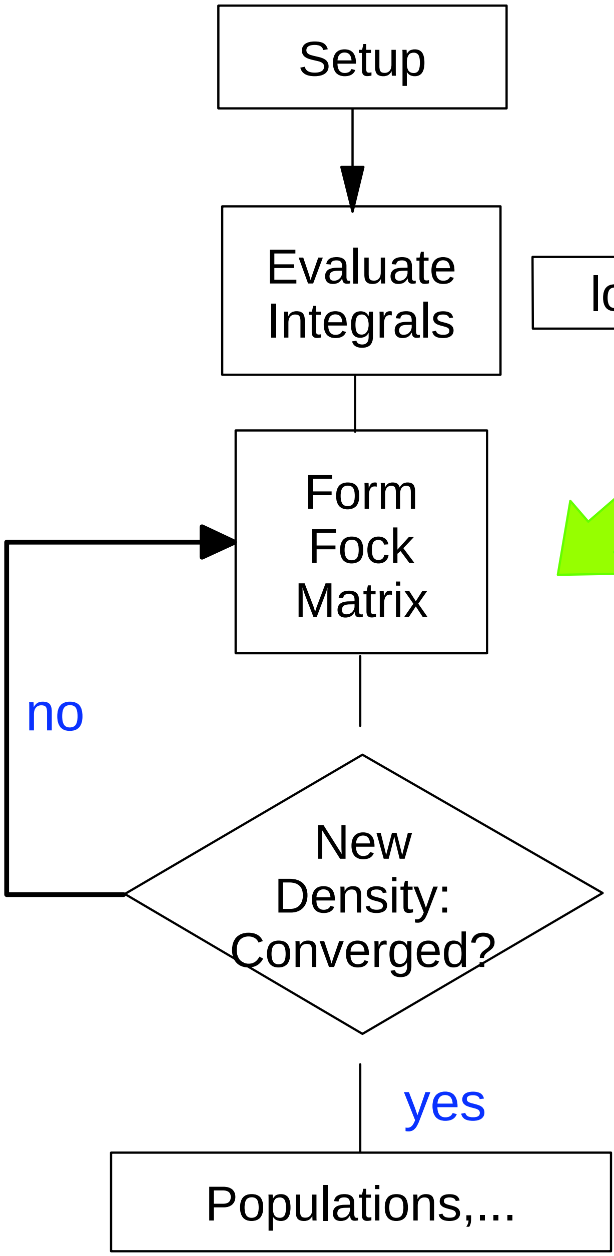


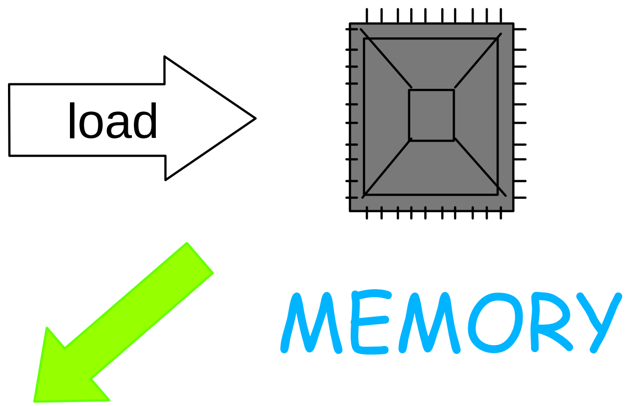
scf=direct
(default)



Incore SCF

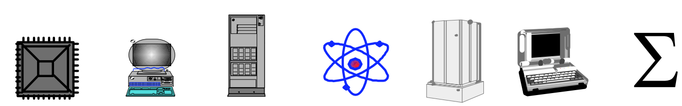
scf=incore





- $N^4/8$ memory ($N^4/4$ for open-shell)
- Fast
- Memory needed:

- 100 basis functions = 100 MB
- 200 basis functions = 1600 MB
- 300 basis functions = 8100 MB



Direct versus Conventional SCF

Almlof and Alrichs: SCF is not N^4!

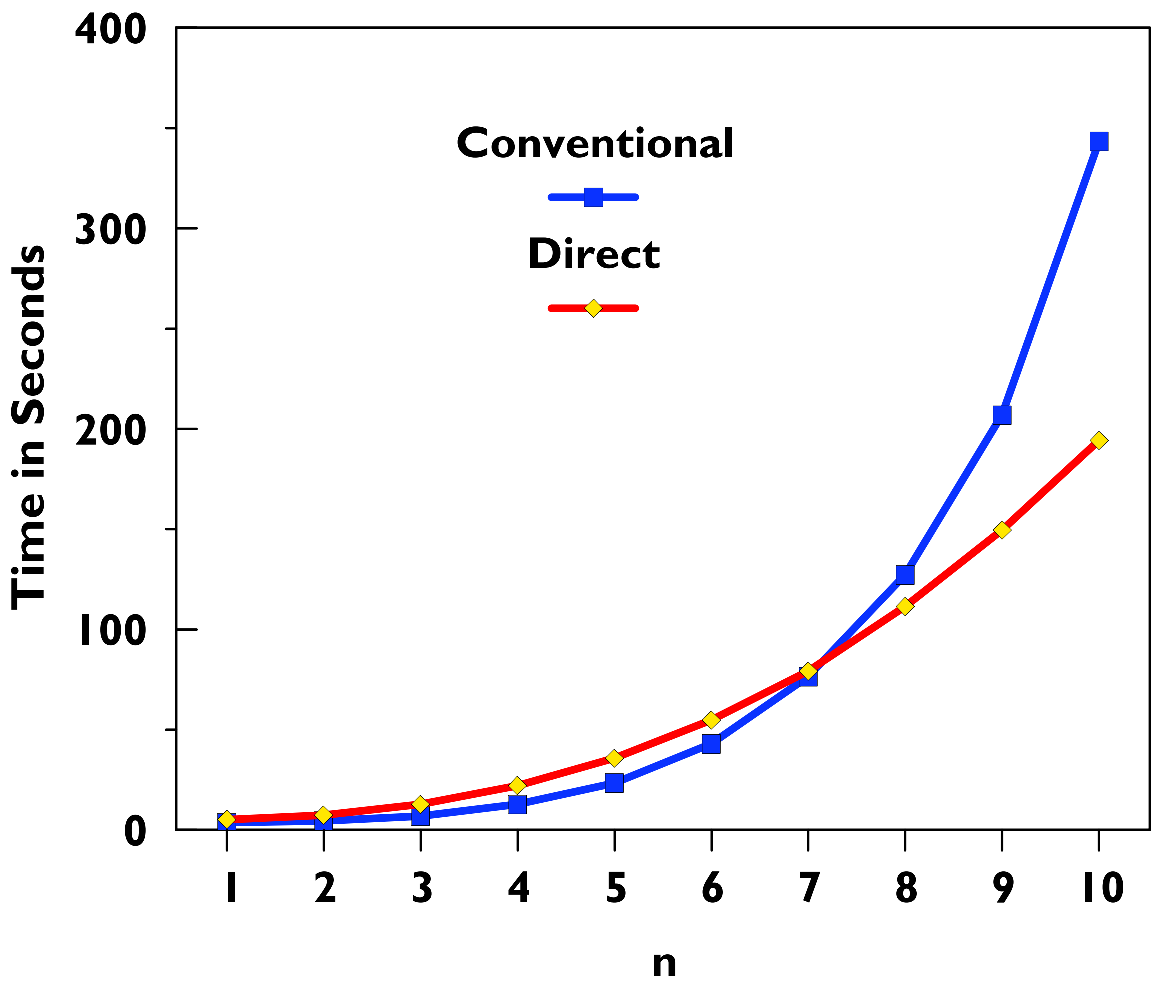
- Direct SCF is faster than conventional for large cases

C_nH_{n+2} Hydrocarbons

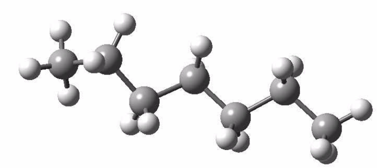
n	Conventional (Sec.)	InCore (Sec.)	Direct (Sec.)	Basis Functions	File Sizes(C)	File Sizes(D)
1	3.8	31.6	5.3	23	22	20
2	4.5	32.4	7.4	42	24	20
3	7.1	34.1	12.8	61	36	20
4	12.8	38.1	22.2	80	66	24
5	23.5	43.7	35.8	99	116	24
6	43.0	52.9	54.8	118	198	24
7	76.5	66.1	79.2	137	312	28
8	127.3	83.7	111.5	156	466	32
9	207.1	-	149.6	175	646	36
10	343.4	-	194.3	194	862	36

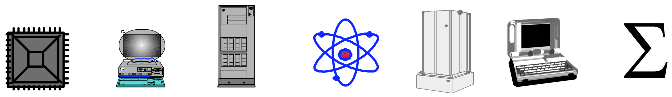
Timings on an IBM WinterHawkII, 375 MHz
Gaussian98 Rev. A10
Incore memory: 900MB
Conv. & Direct memory: 48MB

C_nH_{n+2} Hydrocarbons CPU (C & D)

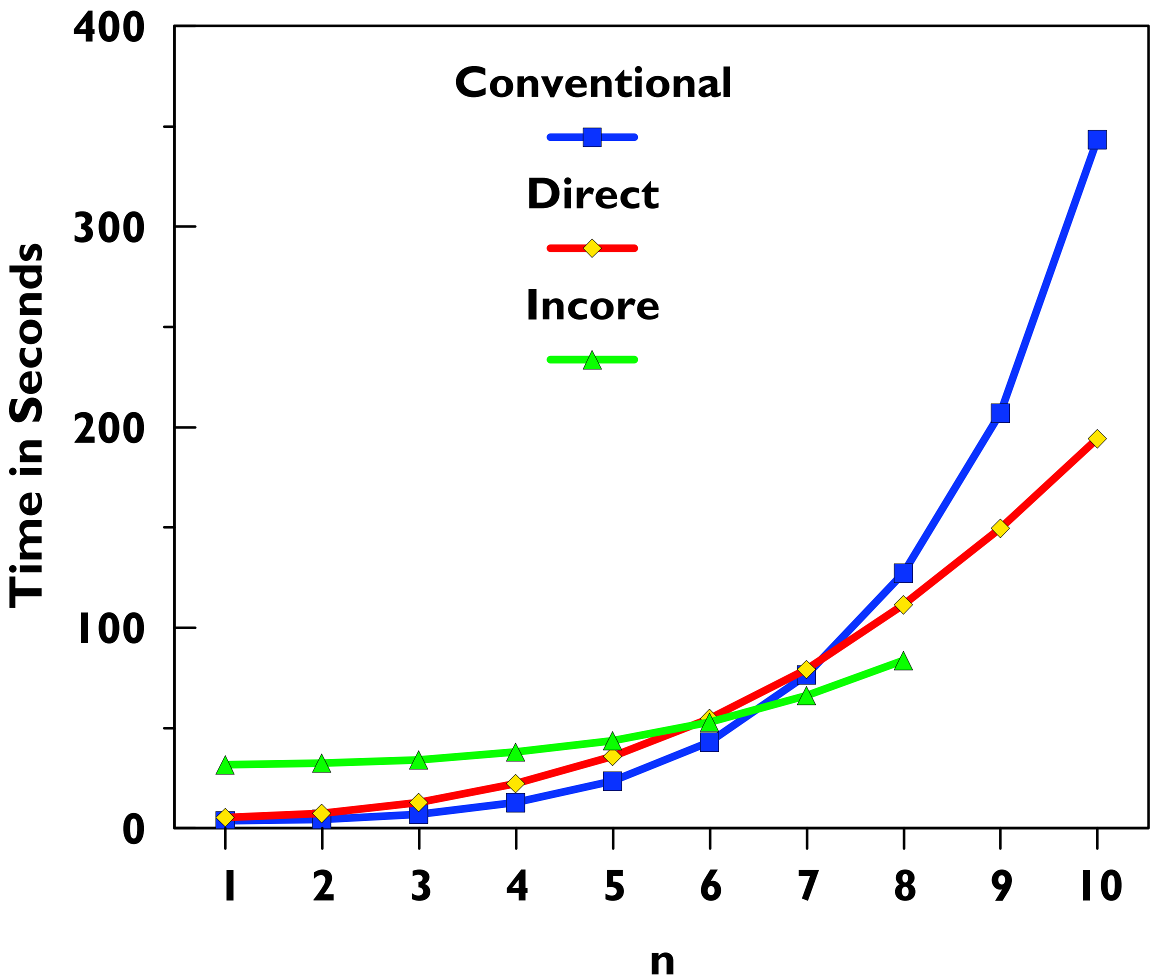


IBM Power3
G98 A.10
C_nH_{n+2}

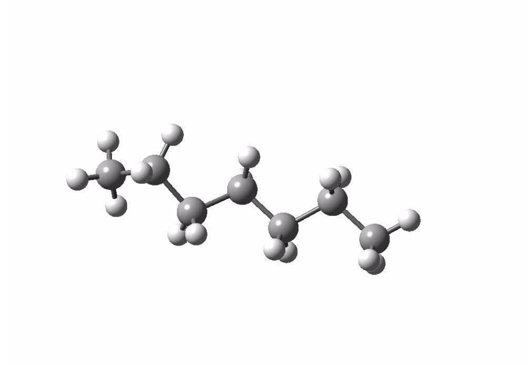


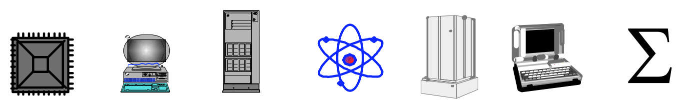


C_nH_{n+2} Hydrocarbons CPU (C, D, & I)

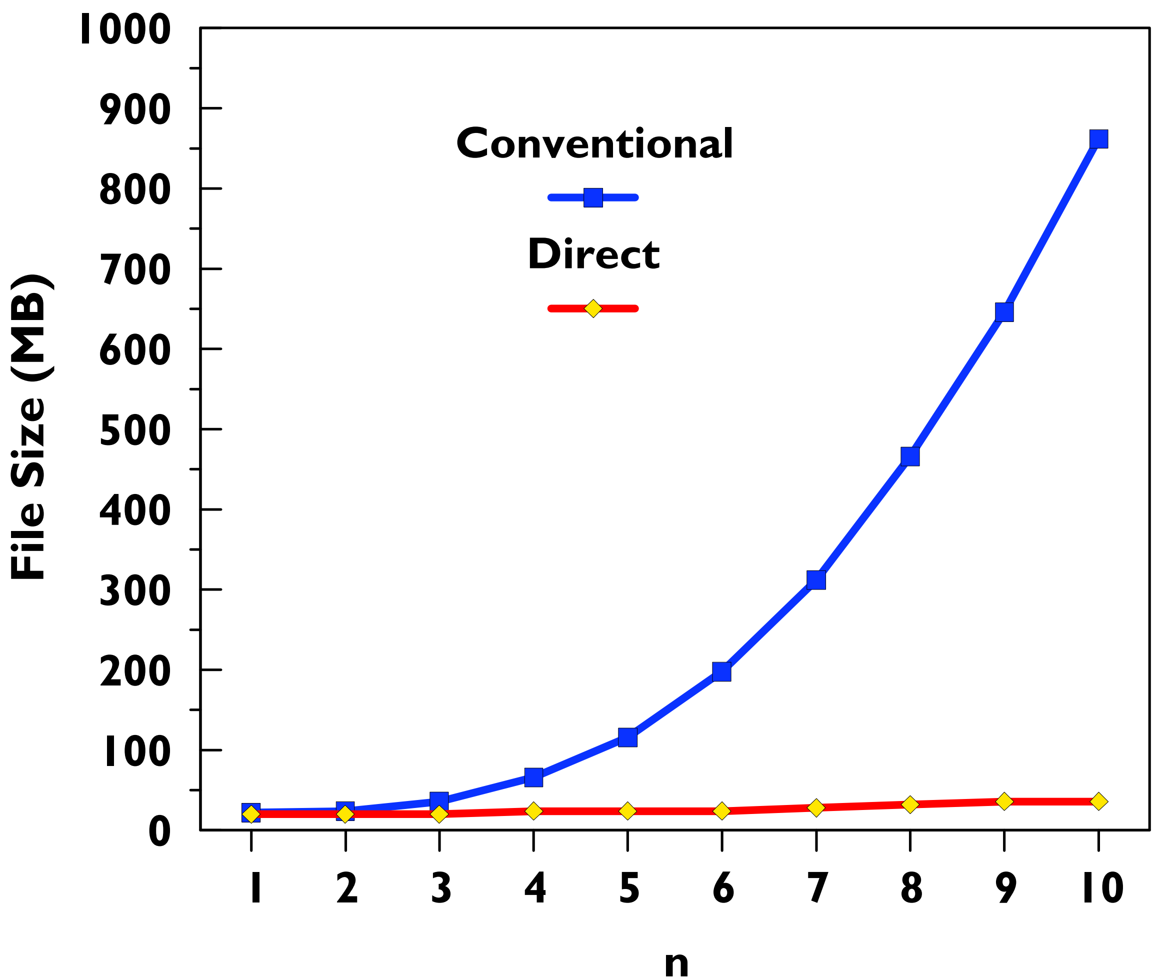


IBM Power3
G98 A.10
C_nH_{n+2}





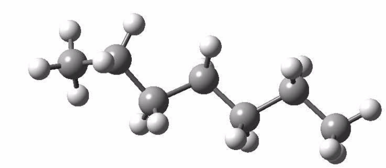
C_nH_{n+2} Hydrocarbons File Sizes

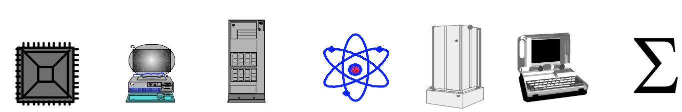


IBM Power3

G98 A.10

C_nH_{n+2}





Hartree-Fock Gradients

$$E_{HF}^{x} = \sum_{\mu\nu} P_{\mu\nu} h_{\mu\nu}^{x} + \frac{1}{2} \sum_{\mu\nu\lambda\sigma} P_{\mu\nu} P_{\lambda\sigma} (\mu\lambda \parallel \nu\sigma) + \sum_{\mu\nu} W_{\mu\nu} S_{\mu\nu}^{x} + V_{nuc}^{x}$$

where,

$$W_{\mu\nu} = -\sum_{\lambda\sigma} P_{\mu\lambda} F_{\lambda\sigma} P_{\sigma\nu}$$

SCF Algorithms

- Direct:
  - $O(N^{2.3})$ CPU
  - Modest memory - 4 MW
  - Faster than conventional
  - Faster than InCore for very large jobs, but can't use InCore for these anyway
- Forces and Optimizations:
  - Integral derivatives can be used as computed
  - No new storage issues
  - Energy + Gradient only 20-30% more CPU than energy

Σ

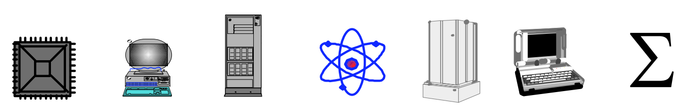
MP2 Energy

$$E_{MP2} = E_{HF} + E^{(2)} = E_{HF} + \frac{1}{4}\sum a_{ij}^{ab}(ij \parallel ab)$$

where,

$$a_{ij}^{ab} = \frac{(ij\|ab)}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}$$

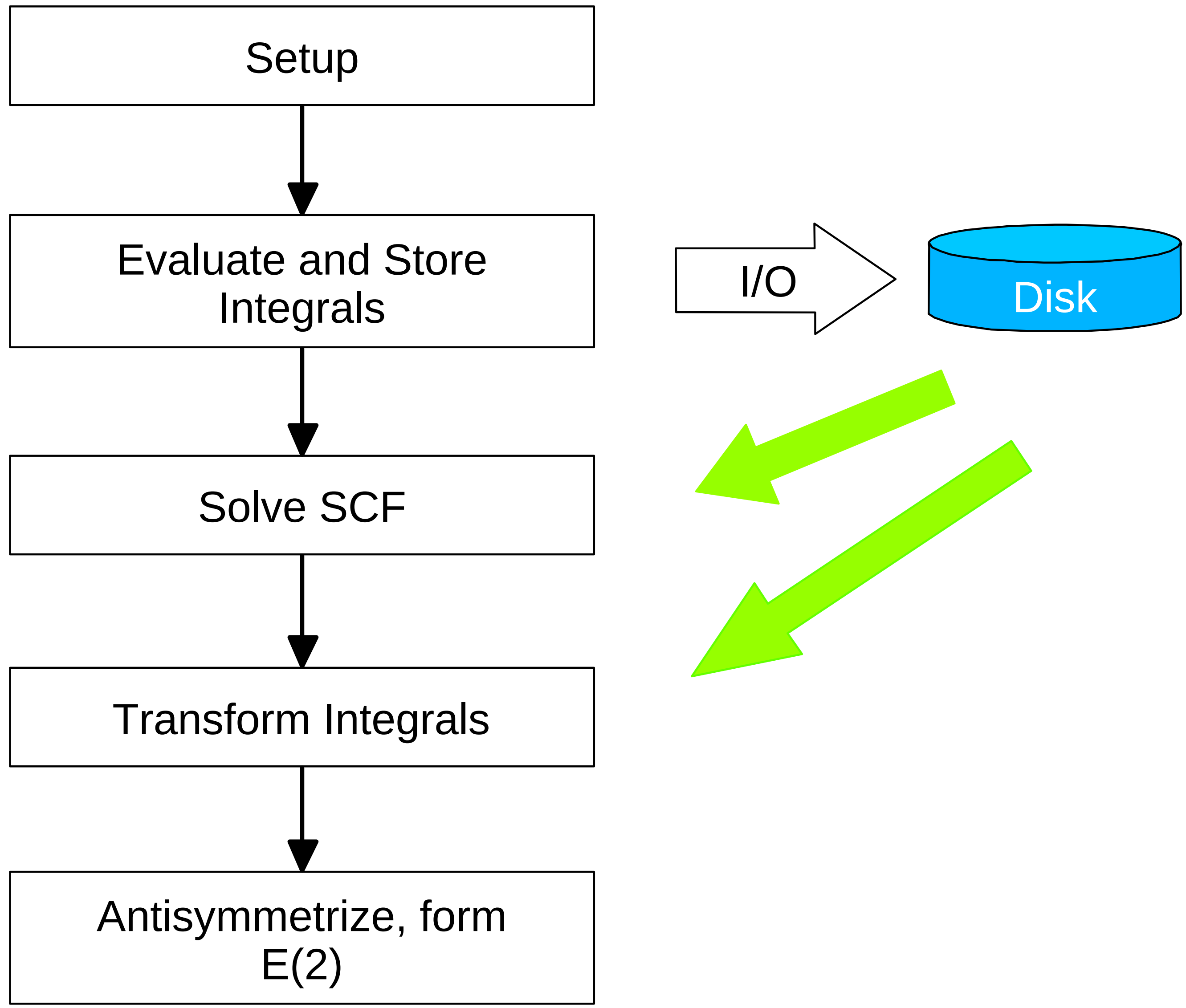
Sum for $E^{(2)}$ is $O(O^2V^2)$, so expensive step is forming (ij||ab)



Traditional MP2 Method

Traditional method: disk-based integral transformation

Conventional MP2 Energy



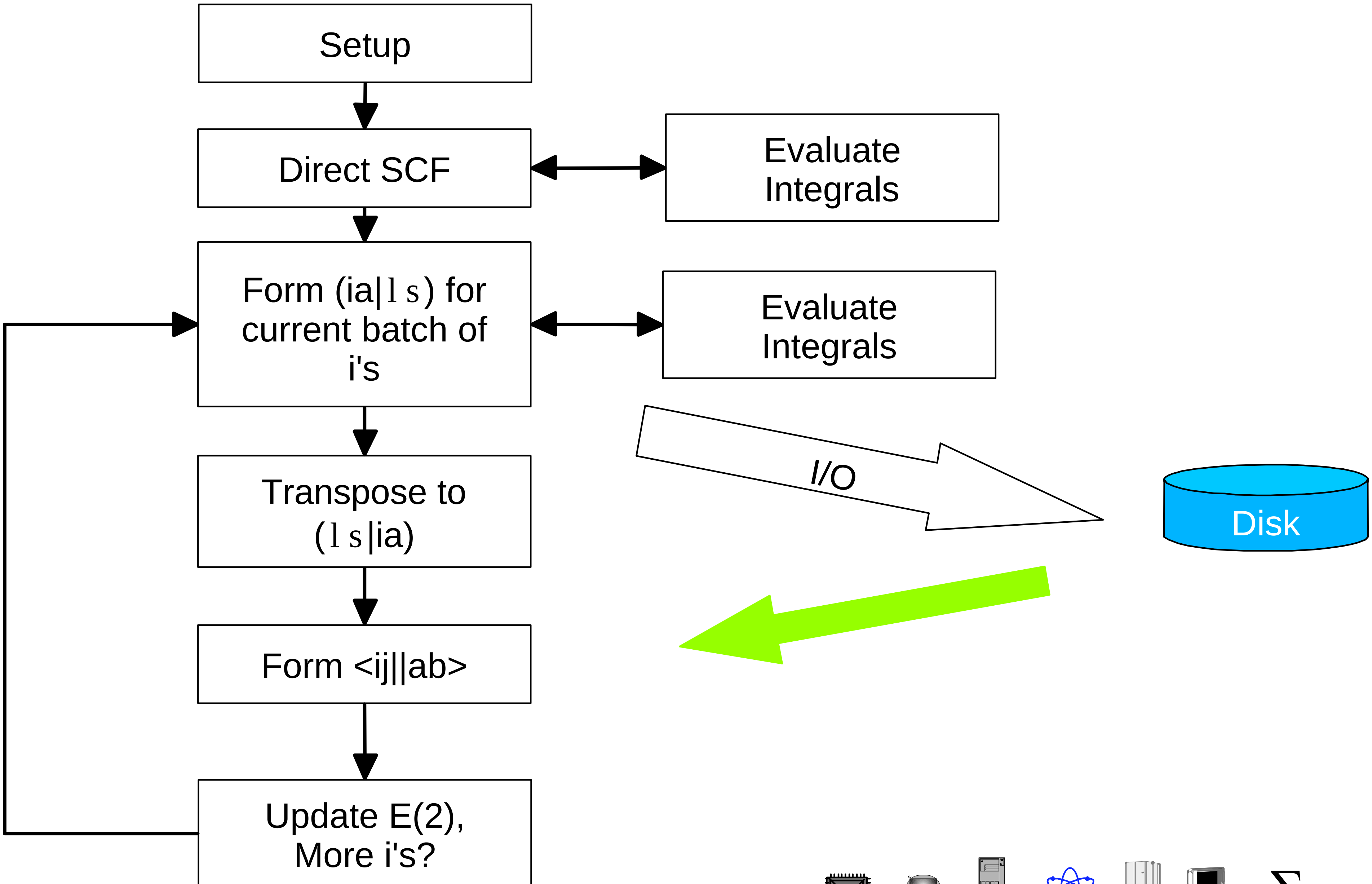
Direct MP2

- Compute integrals while transforming
- Double integral evaluation permits full vectorization
- No external storage or I/O
- OVN memory minimum
- Do O^2VN/Memory integral evaluations, up to O total

Semi-Direct MP2

- Use memory and disk to minimize CPU time
- Sort $(ia|\lambda\sigma)$ into $(\lambda\sigma|ia)$ on disk
- As little as $O(N^2)$ memory and $N^3/2$ disk
- Do $(1/2)OVN^2/\text{MaxDisk}$ integral evaluations
- $OVN^2/2$ disk for one pass

Semi-Direct MP2 Energy



InCore MP2

- Keep AO integrals in main memory
- Need double-length list
- $N^4/4$ memory for closed or open shell

MP2 Gradients

Traditional algorithm:

- Disk for derivatives and energy terms
- I/O time for sorting

Direct algorithm:

- N^3 memory for each i in batch
- Size of system limited by memory

Semi-direct algorithm:

- Almost always preferred
- Minimum $O(N^2)$ memory, $N^3/2$ disk
- 6-8 MW for spdf

Σ

MP2 Frequencies

- Only semi-direct algorithm
- 8MW for f functions, 12MW for g functions
- MP2=stingy option default for better disk re-use
- MP2=NoStingy uses more disk, is slightly faster
- Minimum disk

- $N^4/4+OVN^2/2$ words
- MaxDisk obeyed
- Tries calculation in minimum disk regardless

C_nH_{n+2} Hydrocarbons-MP2 Calculations

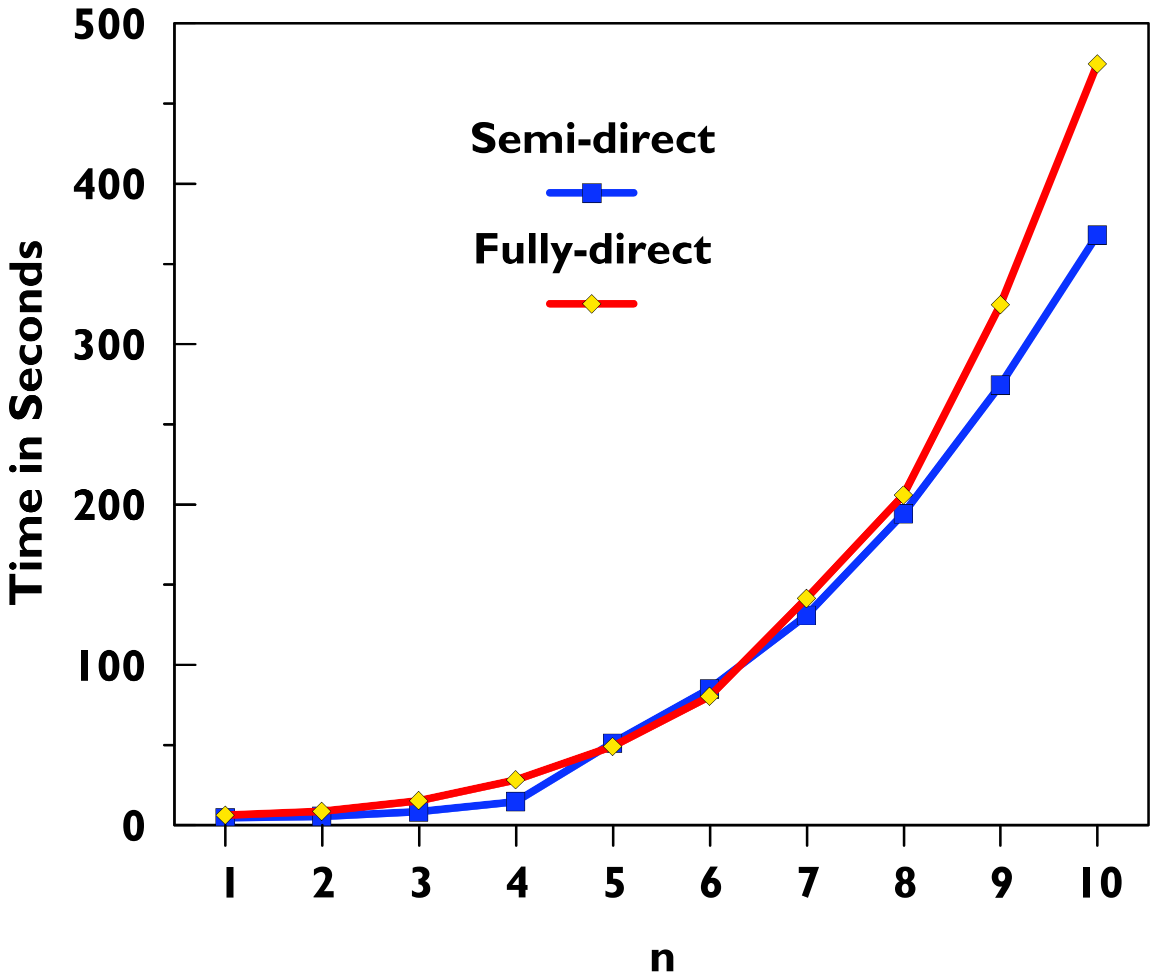
n	Semi-direct (Sec.)	Fully-direct (Sec.)	Basis Functions	File Sizes(SD)	File Sizes(FD)
1	4.7	6.2	23	28	20
2	5.6	8.7	42	29	20
3	8.5	15.5	61	42	20
4	14.9	28.4	80	71	24
5	51.3	49.2	99	126	24
6	85.0	80.4	118	221	24
7	130.9	141.6	137	361	28
8	194.3	205.9	156	548	32
9	274.5	324.5	175	779	36
10	368.0	474.7	194	1072	36

Timings on an IBM WinterHawkII, 375 MHz

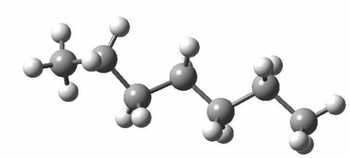
Gaussian98 Rev. A10

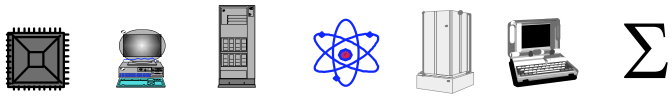
Semi-direct & Fully Direct memory: 48MB

C_nH_{n+2} Hydrocarbons CPU (SD & FD)

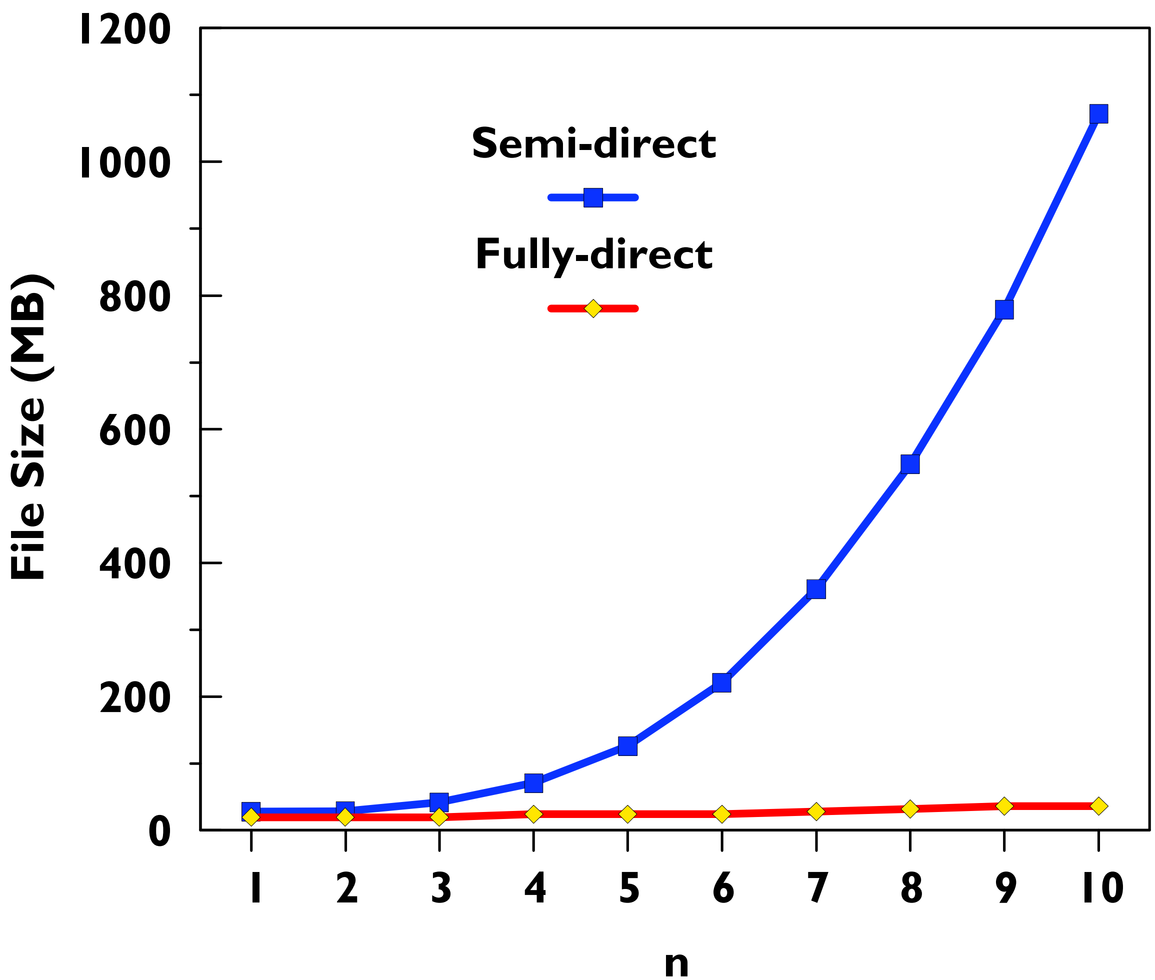


IBM Power3
G98 A.10
C_nH_{n+2}





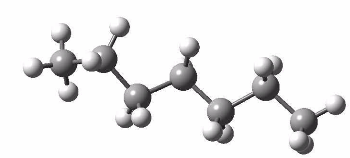
C_nH_{n+2} Hydrocarbons File Sizes-MP2



IBM Power3

G98 A.10

C_nH_{n+2}



MP2 - Frequency

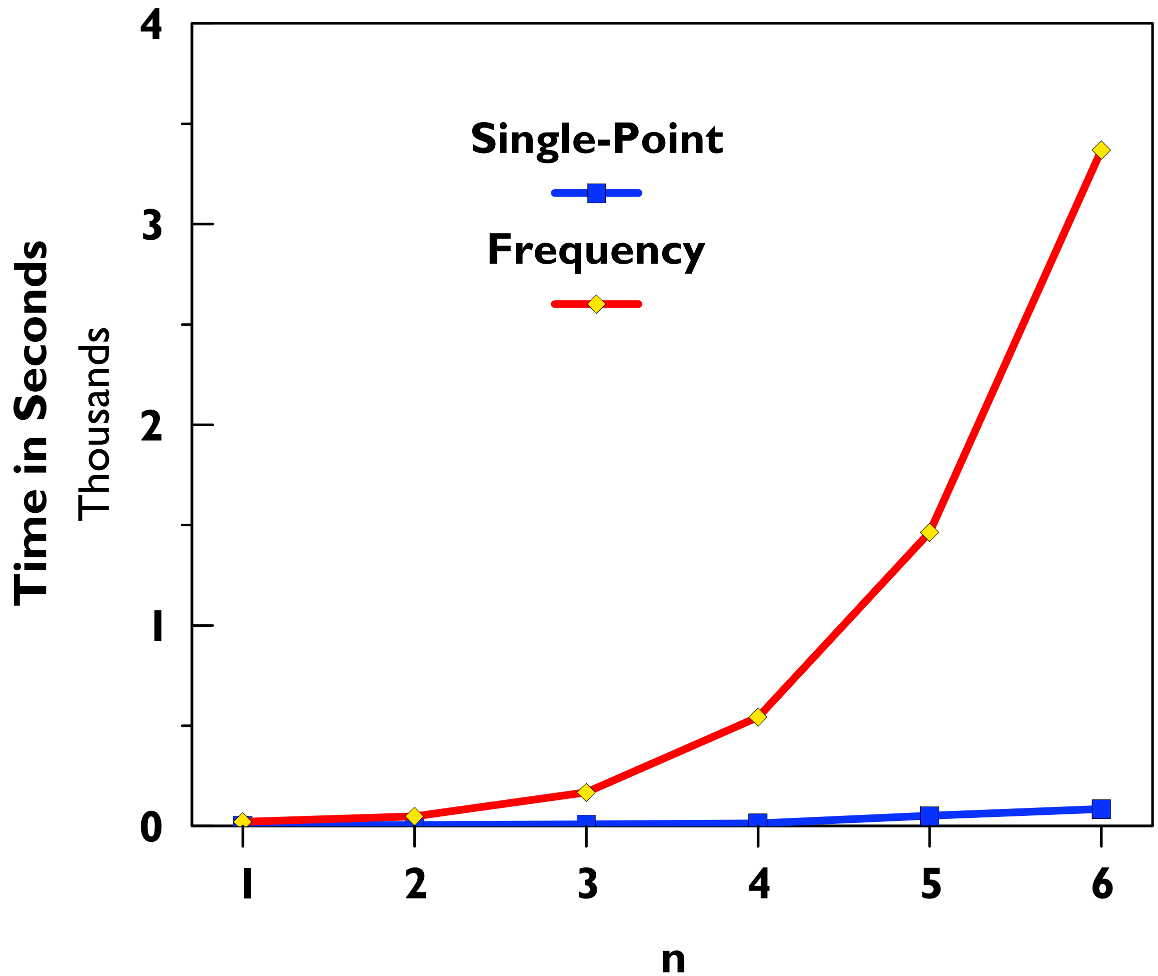
n	Single-Point (Sec.)	Frequency (Sec.)	Basis Functions	File Sizes(SP)	File Sizes(F)
1	4.7	22.1	23	28	28
2	5.6	48.0	42	29	53
3	8.5	169.1	61	42	143
4	14.9	544.0	80	71	351
5	51.3	1464.6	99	126	751
6	85.0	3369.7	118	221	1435

Timings on an IBM WinterHawkII, 375 MHz

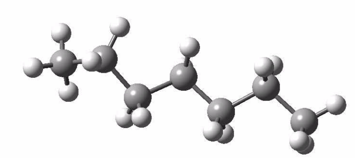
Gaussian98 Rev. A10

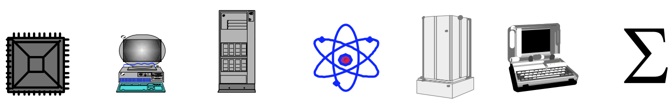
Semi-direct & Fully Direct memory: 48MB

MP2 Frequency & SP CPU Comparison

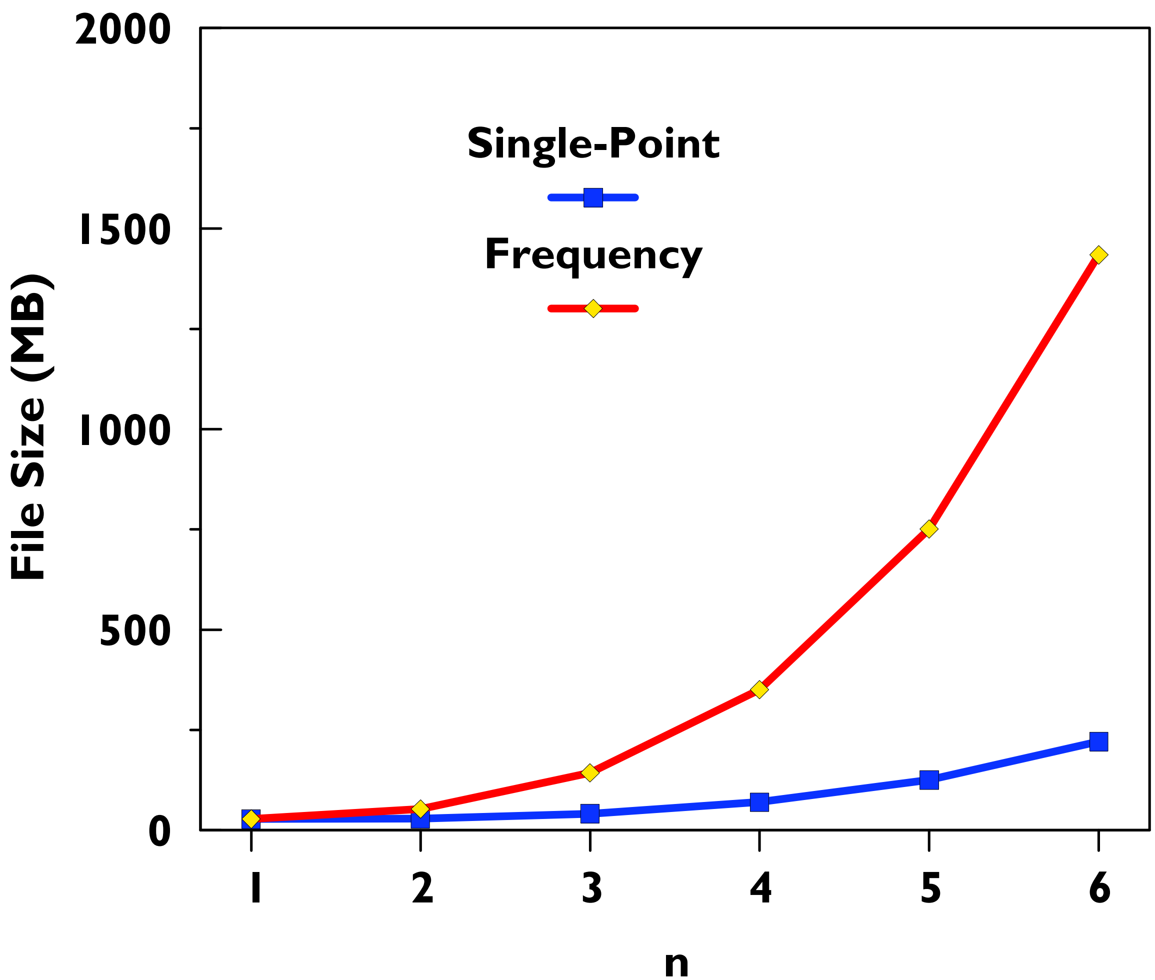


IBM Power3
G98 A.10
C_nH_{n+2}





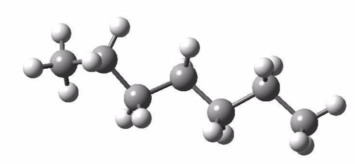
MP2 Freq & SP Disk Usage Comparison

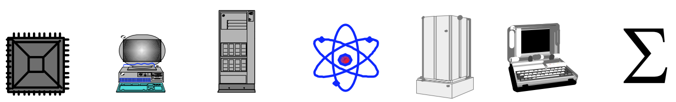


IBM Power3

G98 A.10

C_nH_{n+2}





Integral Transformation

Traditionally used for everything aftre SCF:

$$(pq \mid rs) = \sum_{\sigma} C_{\sigma s} \sum_{\lambda} C_{\lambda r} \sum_{\nu} C_{\nu q} \sum_{\mu} C_{\mu p} (\mu \nu \mid \lambda \sigma)$$

Gaussian uses semi-direct algorithm:

- Fixed minimum memory for integral evaluation
- Better behavior for large systems and limited memor
- Generate <*pq*||*rs*> during transformation
- Can make <*ij*||*ab*> using only $O(O^2N^2)$ disk

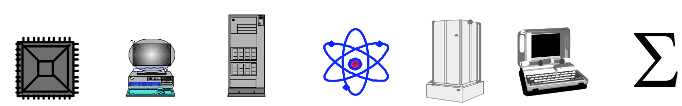
MAXDISK

Specifies the amount of disk storage available for scratch data, in 8-bytes words

The units can be: KB, MB, GB, KW, MW or GW

example:

maxdisk=8MB



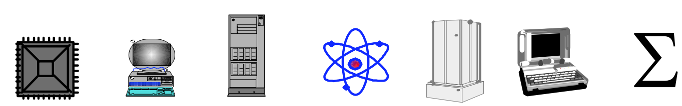
Size Dependence of Methods

Method	Formal CPU	Formal Memory	Formal Disk	Actual CPU	Actual Disk
Conv. SCF	N^4	N^2	N^4	$N^{3.5}$	$N^{3.5}$
Incore SCF	N^4	N^4	-	N^4	N^2
Direct SCF	N^4	N^2	-	$N^{2.3}$	N^2
Conv. MP2	ON^4	N^2	N^4	ON^4	N^4
Dir MP2 SP	ON^4	OVN	-	O^2N^3	N^2
SD MP2 SP	ON^4	N^2	VN^2	O^2N^3	VN^2
Conv. MP2 Force	ON^4	N^2	N^4	ON^4	N^4
Dir MP2 Force	ON^4	N^3	-	O^2N^3	N^2
SD MP2 Force	ON^4	N^2	N^3	O^2N^3	N^3
MP3, CISD, QCISD	O^2N^4	N^2	N^4	O^2N^4	N^4
MP4, QCISD(T)	O^3V^4	N^2	N^4	O^3V^4	N^4

O: Number of occupied orbitals

V: Number of virtual orbitals

N: Number of basis functions



Parallel Gaussian

Efficiency Considerations

Amdahl's Law

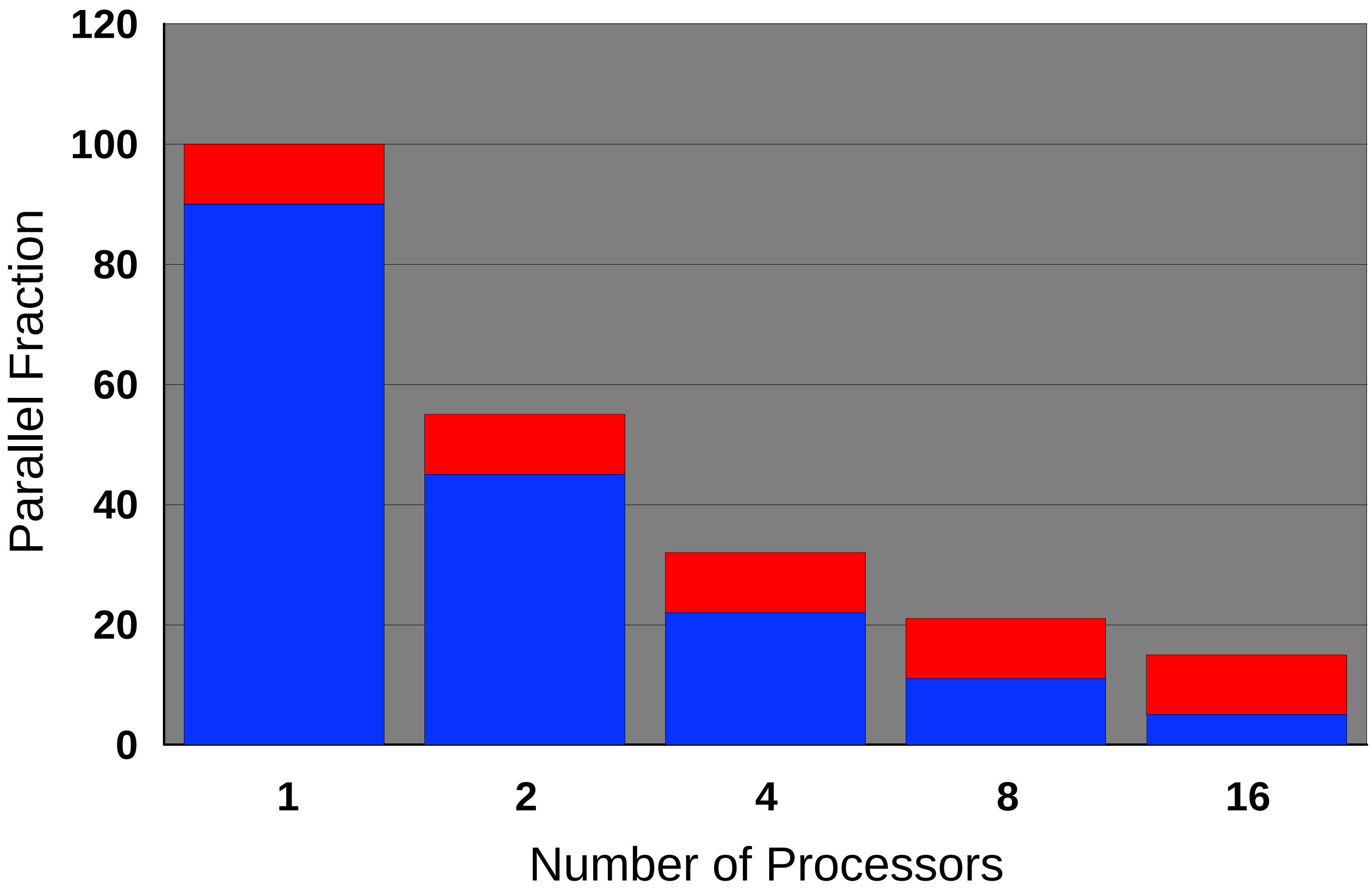
Although a code contains parallel constructs, the serial processing in the code will dominate its overall performance

To estimate expected parallel speedups:

$$S(N) = \frac{1}{f_S + \frac{f_P}{N}}$$

$S(N)$ Maximun expected speedup from parallelization
N Number of processors available for parallel execution
f_P Fraction of a program that can execute in parallel
f_S Fraction of a program that is serial = $1 - f_P$

Amdahl's Law Example



Estimating Memory Requirements

Single processor memory requirement = M + 2N^2

M= Required value for a job type
N= Number of Basis Functions

	f functions	g functions	h functions	i functions	j functions
SCF Energies	~6 MW	~6 MW	~12 MW	~25 MW	~64 MW
SCF Gradients	~6 MW	~7 MW	~19 MW	~40 MW	
SCF Frequencies	~6 MW	~11 MW	~30 MW		
MP2 Energies	~6 MW	~7 MW	~14 MW	~30 MW	~74 MW
MP2 Gradients	~6 MW	~8 MW	~18 MW	~40 MW	
MP2 Frequencies	~8 MW	~12 MW	~30 MW		

1 MW = 1,048,576 Words = 8,388,608 bytes

Example: 300 basis functions HF geometry optimization using g functions would require about 7.2 MW (~60MB)

Σ

FreqMem Utility

- FreqMem utility:
  - Returns minimum memory size for optimal performance
- Example:
  - **freqmem N_A N R/U C/D SP/SPD/SPDF**

N_A = number of atoms

N = number of basis functions

R/U = restricted/unrestricted

C/D = conventional/direct

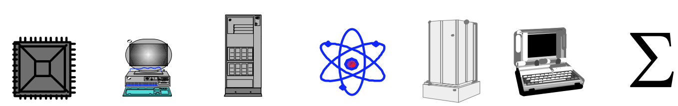
SP/SPD/SPDF = functions in basis set

Memory Allocation Empirical Formula

Parallel calculations with more that one processor on shared-memory systems require additional memory

total_mem = sp_mem + (n - 1) * 0.75 * sp_mem

total_mem = total memory required for the parallel run
sp_mem = single processor memory required
n = number of processors



Parallel SCF

```
*Deck PRSMsu
     subroutine PRSMsu

       loop over Nprocessors

        call PRISM

       end loop

       loop over Nprocessors ( serial code )

         add 1/Nprocessors Fock Matrix contributions

       end loop
```

Parallel Speedup & Efficiency

Speedup (S) is defined as the ratio of the serial run time (elapsed, t_s) over the time that it takes to do the same problem in parallel (elapsed time, t_p)

$$S = \frac{t_s}{t_p}$$

$$e = \frac{S}{N_{processors}}$$

Extrapolated Speedup

$$s = \frac{1}{\left(\frac{p}{N_{processors}}\right) + (1 - p)}$$

$$p = \frac{S_{N_{processors}} - S_{M_{processors}}}{(1 - 1/N_{processors}) \times S_{N_{processors}} - (1 - 1/M_{processors}) \times S_{M_{processors}}}$$

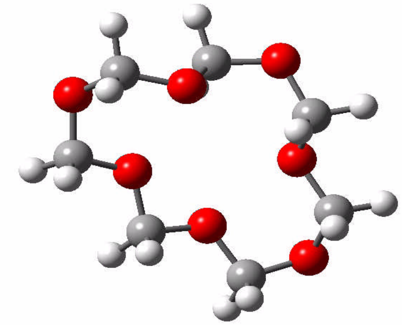
Parallel Links in Gaussian98

Link	Description
L302	Overlap integrals
L303	One-electron properties integrals
L502	Closed-and open-shell SCF solution
L506	GVB solution
L508	Quadratically convergent SCF solution
L510	Multiconfiguration SCF solution
L602	One-electron properties
L703	Two-electron integral first or second derivative evaluation
L906	Direct and semi-direct MP2 energies and gradients
L914	Calculates excited states using CI with single excitations
L1002	CPHF soution and contribution of coefficient derivatives to Hartree-Fock second derivatives
L1014	Coupled perturbed CI singles
L1110	Two-electron contributions to Fock matrix derivatives with respect to nuclear coordinates
L1112	Forms most of the terms in MP2 second derivatives

Linda links

Crown ether Example

Processors	Elapsed Time (Sec)[3]	Speedup
1	4549	1
2		
ethernet[1]	2271	2
switch[1]	2268	2
shared-memory[2]	2365	2
8		
ethernet[1]	652	7
switch[1]	610	7
shared-memory[2]	626	7
16		
ethernet[1]	442	10
switch[1]	372	12
shared-memory[2]	386	12

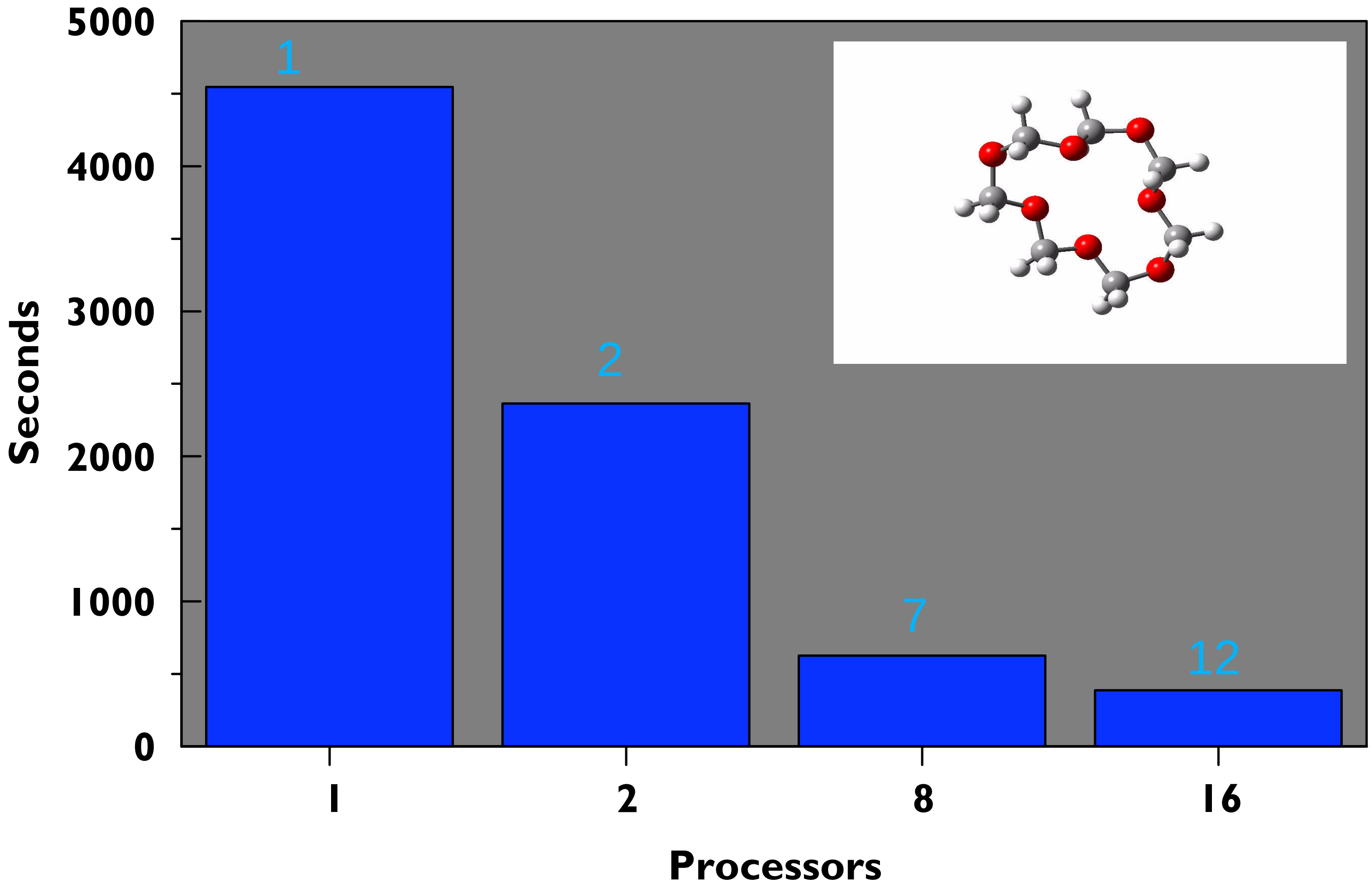


$(OCH_2)_7$, Crown ether
HF/6-31G* FOPT OPTCYC=3

[1] 16X(4-way nodes), Power3-II, 375 MHz, 8MB L2
[2] 1X(16-way node), Power3-II, 375 MHz, 8MB L2
[3] *Gaussian98* Rev. A.7, xlf 5.1.1 Compiler

Crown ether Parallel Speedup

Speedup



$(OCH_2)_7$, Crown ether
HF/6-31G* FOPT OPTCYC

Σ

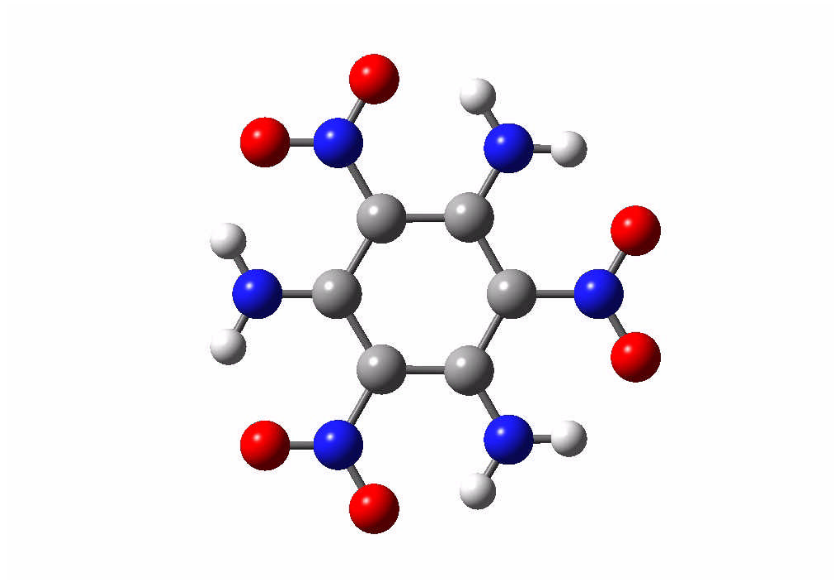
test178

test178: RHF/6-31G** SCF=DIRECT POP=NPA PROP=FIT

300 Basis Functions

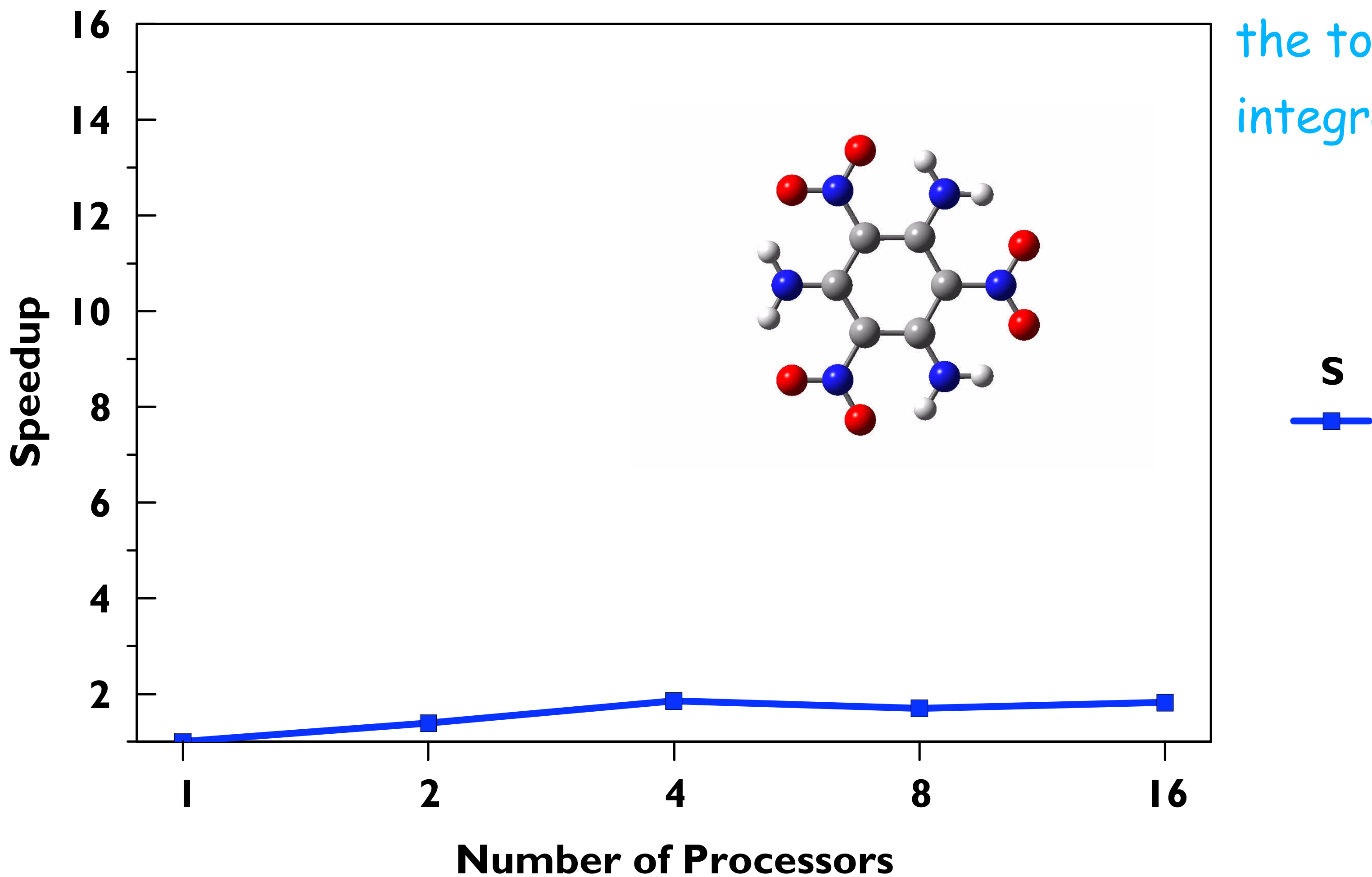
Full Point Group D_{3H}

Processor	Time	Speedup
1	153.39	1.00
2	109.86	1.39
4	82.90	1.85
8	90.01	1.70
16	84.48	1.82



Gaussian 98 Rev. A.7
Shared-memory

test178 Scalability



Symmetry reduces the total number of integrals

test178: RHF/6-31G** SCF=DIRECT POP=NPA PROP=FIT

300 Basis Functions

Full Point Group D_{3H}

a -pinene SP Scalability

Porcessors	Time	Speedup
1		
HF	2880.54	1.00
B3-LYP	4022.71	1.00
2		
HF	1463.07	1.97
B3-LYP	2036.24	1.98
4		
HF	726.13	3.97
B3-LYP	1031.06	3.90
8		
HF	360.90	7.98
B3-LYP	515.95	7.80
16		
HF	194.39	14.82
B3-LYP	285.37	14.10

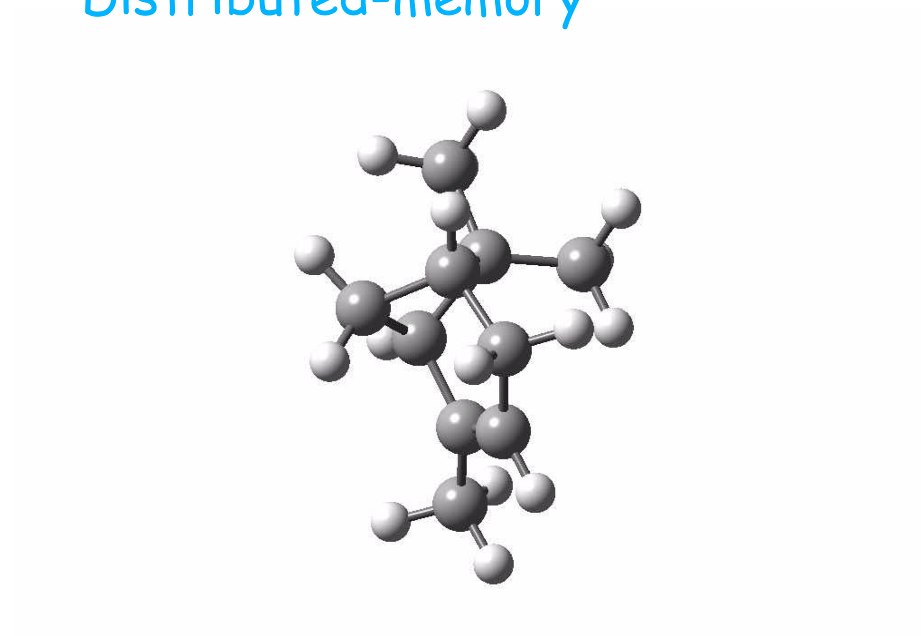
α-pinene HF/6-311G(df,p) &

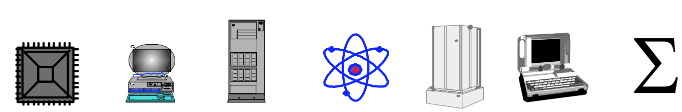
B3-LYP/6-31G(df,p)

346 Basis Functions

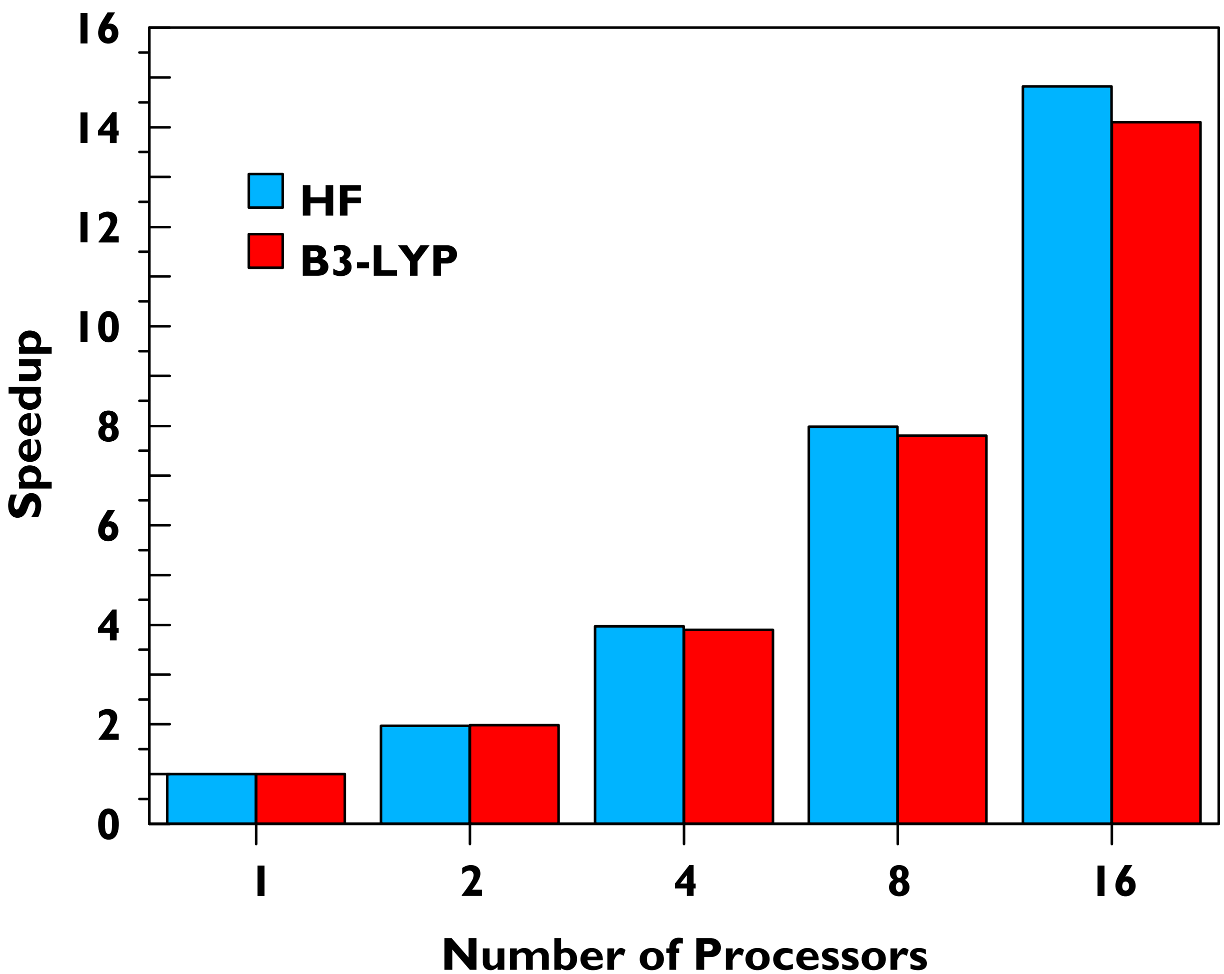
$C_{10}H_{16}$

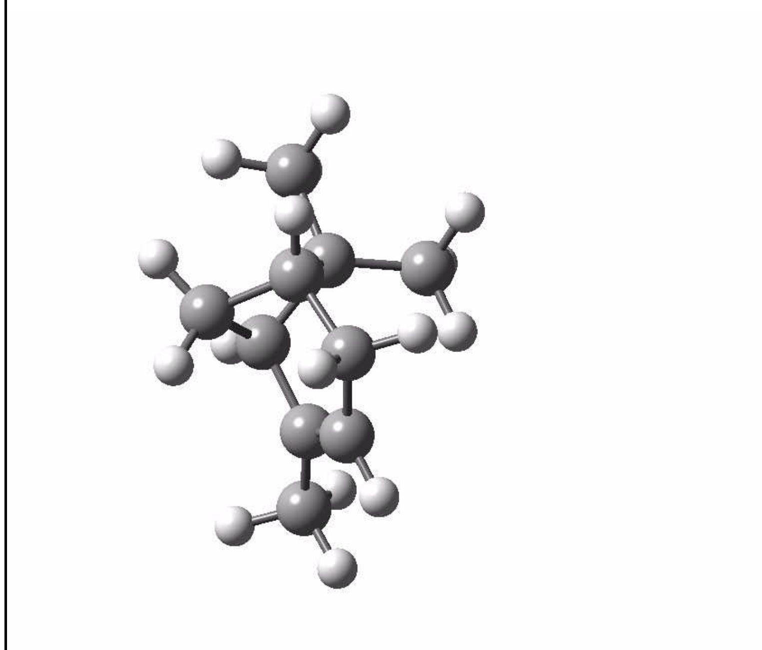
Distributed-memory



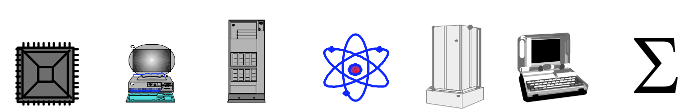


a-pinene: Hf & DFT Scalability



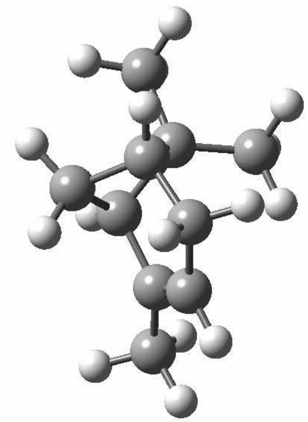


HF/6-311G(df,p) & B3-LYP/6-31G(df,p)
346 Basis Functions
$C_{10}H_{16}$
Distributed-memoy



a -pinene Frequency Calculation

B3-LYP/6-31G* FREQ
182 Basis Functions
G98 Rev. A.7
shared-memory

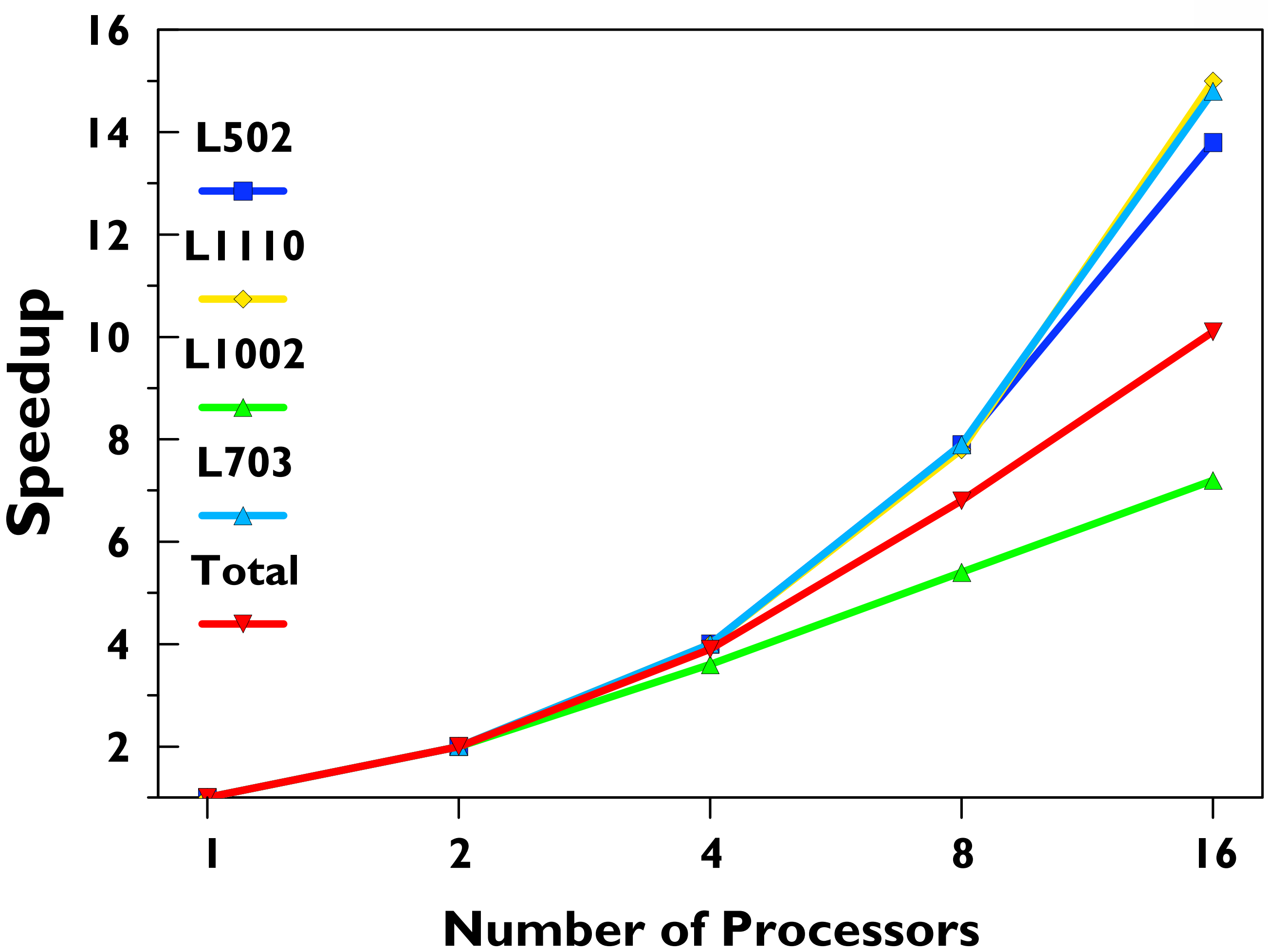


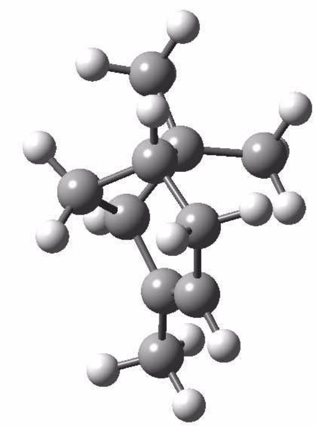
Time in Sec.

Processors	L502	S[a]	L1110	S[a]	L1002	S[a]	L703	S[a]	Total	S[a]
1	1076	1.0	2802	1.0	3144	1.0	3702	1.0	10738	1.0
2	515	2.0	1402	2.0	1602	2.0	1827	2.0	5362	2.0
4	254	4.0	700	4.0	879	3.6	911	4.0	2764	3.9
8	136	7.9	359	7.8	580	5.4	471	7.9	1576	6.8
16	78	13.8	187	15.0	437	7.2	251	14.8	998	10.1

[a] Speedup

a -pinene Speedups

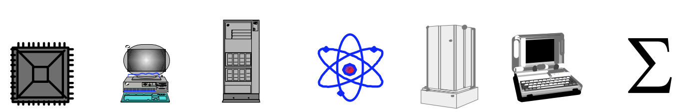




B3-LYP/6-31G*

Frequency

182 Basis Functions



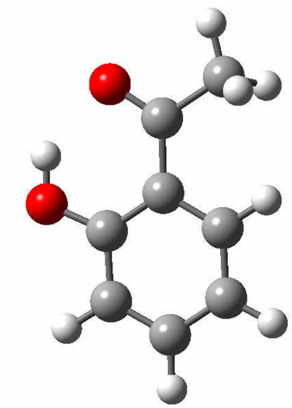
CIS Calculation

CIS=direct, 6-31++G, scf=direct, force

154 Basis Functions

Distributed-memory

G98 Rev. A.7

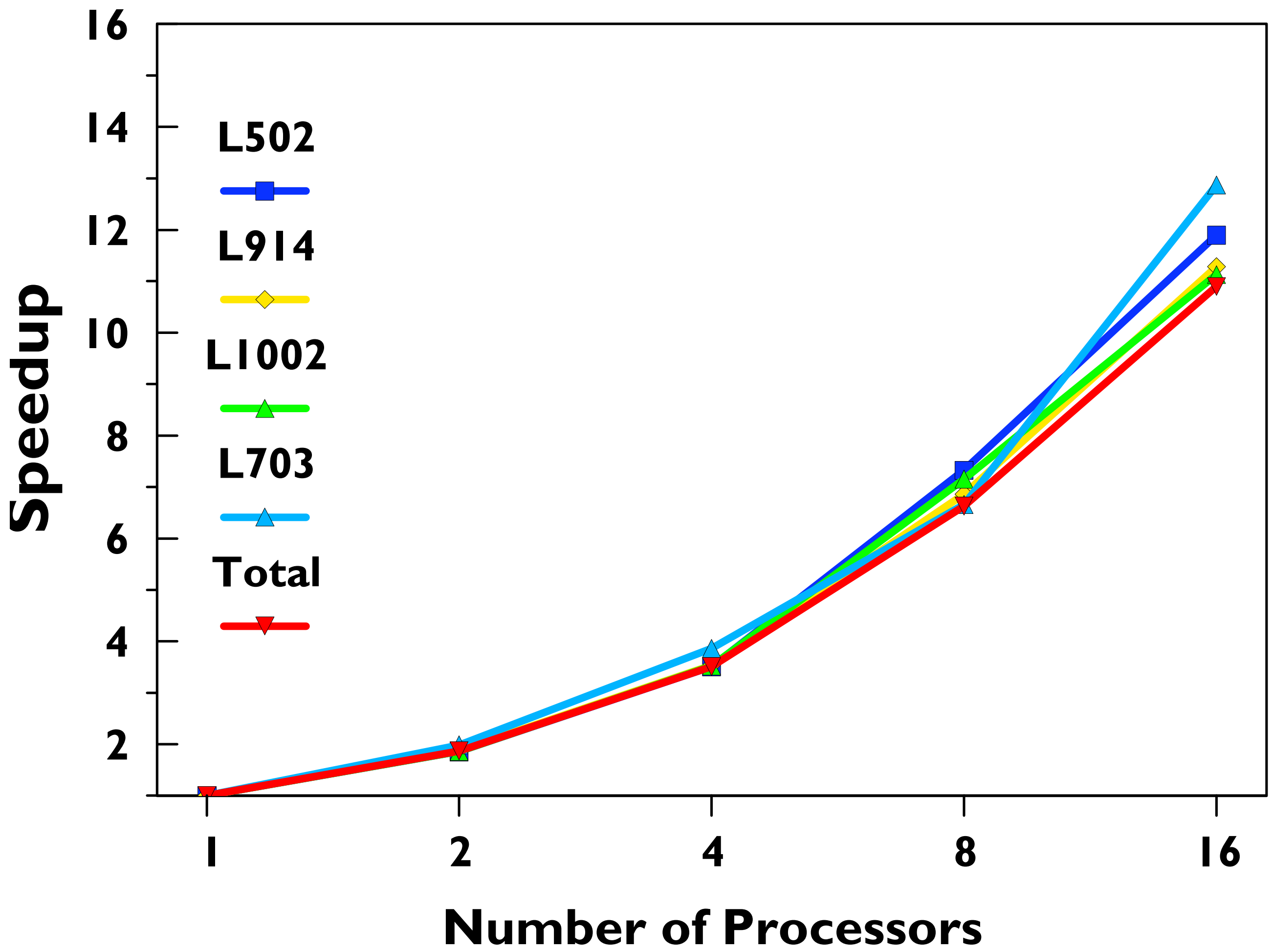


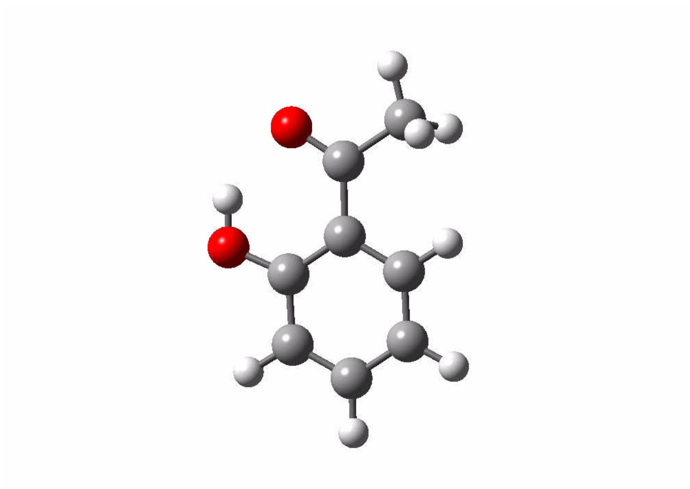
Processors	L502	S[a]	L914	S[a]	L1002	S[a]	L703	S[a]	Total	S[a]
1	821	1.00	1455	1.00	701	1.00	193	1.00	3182	1.00
2	441	1.86	776	1.88	377	1.86	97	1.99	1703	1.87
4	234	3.51	411	3.54	199	3.52	50	3.86	906	3.51
8[b]	112	7.33	212	6.86	98	7.15	29	6.66	480	6.63
16	69	11.90	129	11.28	63	11.13	15	12.87	292	10.90

[a] Speedup

[b] Shared-memory

CIS Scalability



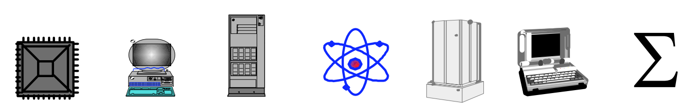


CIS=direct, 6-31++G, scf=direc force

154 Basis Functions

Distributed-memory

G98 Rev. A.7



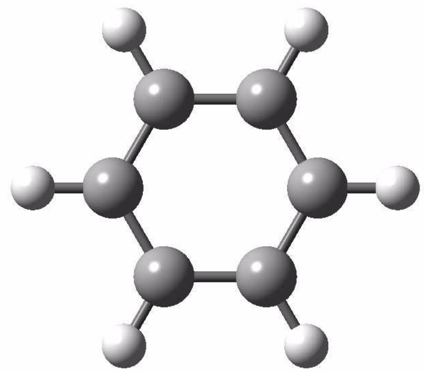
MCSCF Calculation

CAS(6,6), 6-31+G(3df), guess=cards, NOSYM

240 Basis Functions

Shared-memory

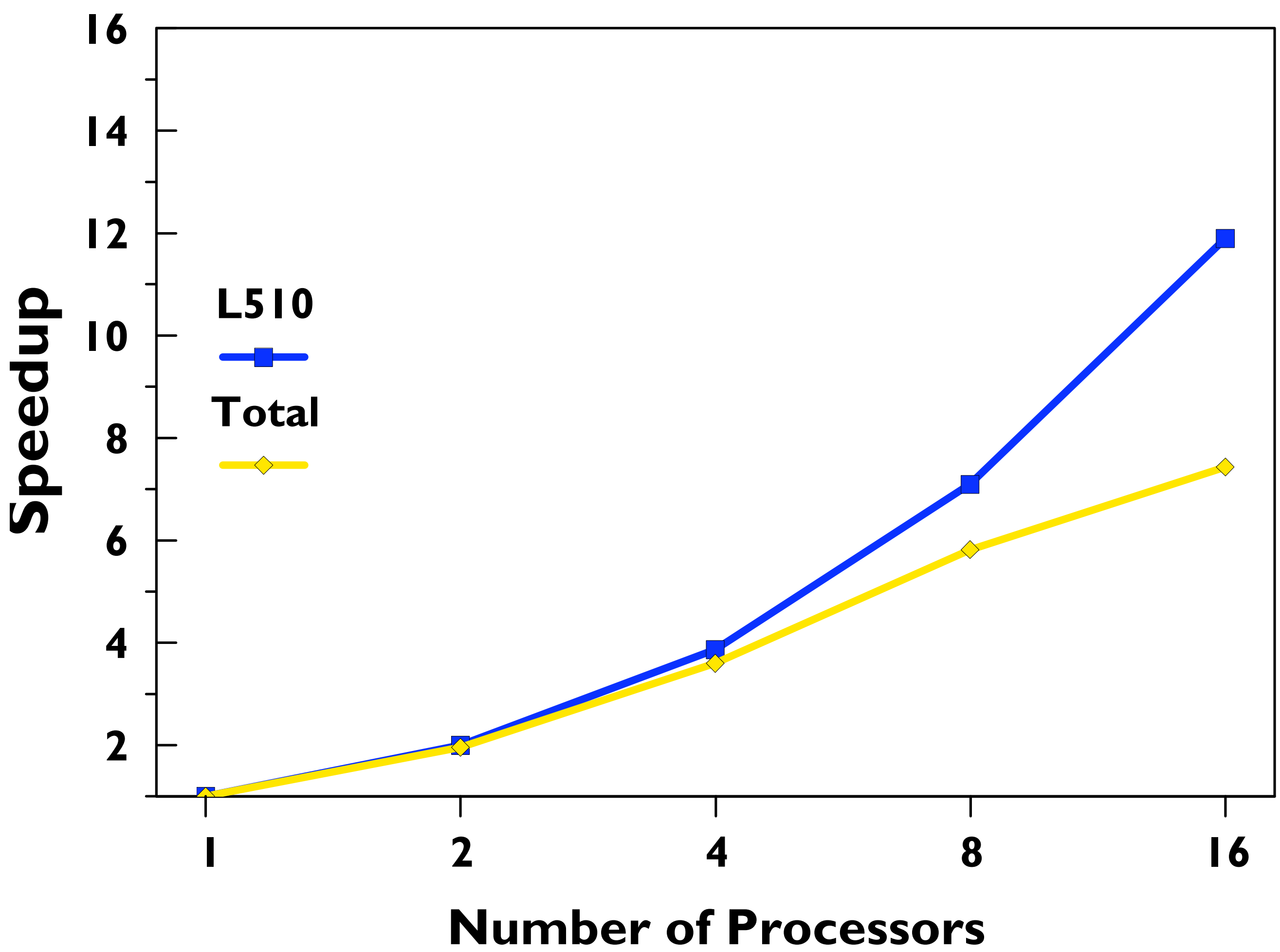
G98 Rev. A.7

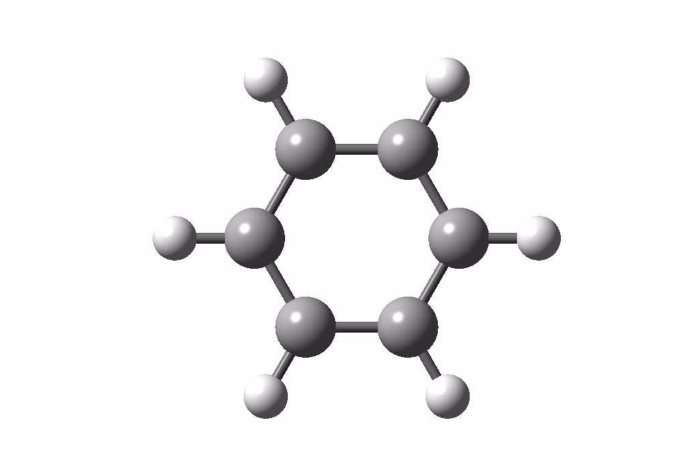


Processors	L510	S[a]	Total	S[a]
1	476	1.00	483	1.00
2	238	2.00	246	1.96
4	123	3.87	134	3.60
8	67	7.10	83	5.82
16	40	11.90	65	7.43

Σ

MCSCF Scalability





CAS(6,6), 6-31+G(3df),
guess=cards, NOSYM
240 Basis Functions
Shared-memory
G98 Rev. A.7

Summary (for an n-way system)

HF: SP, Gradients, Freq

DFT: SP, Gradients, Freq

CIS: SP, Gradients, Freq

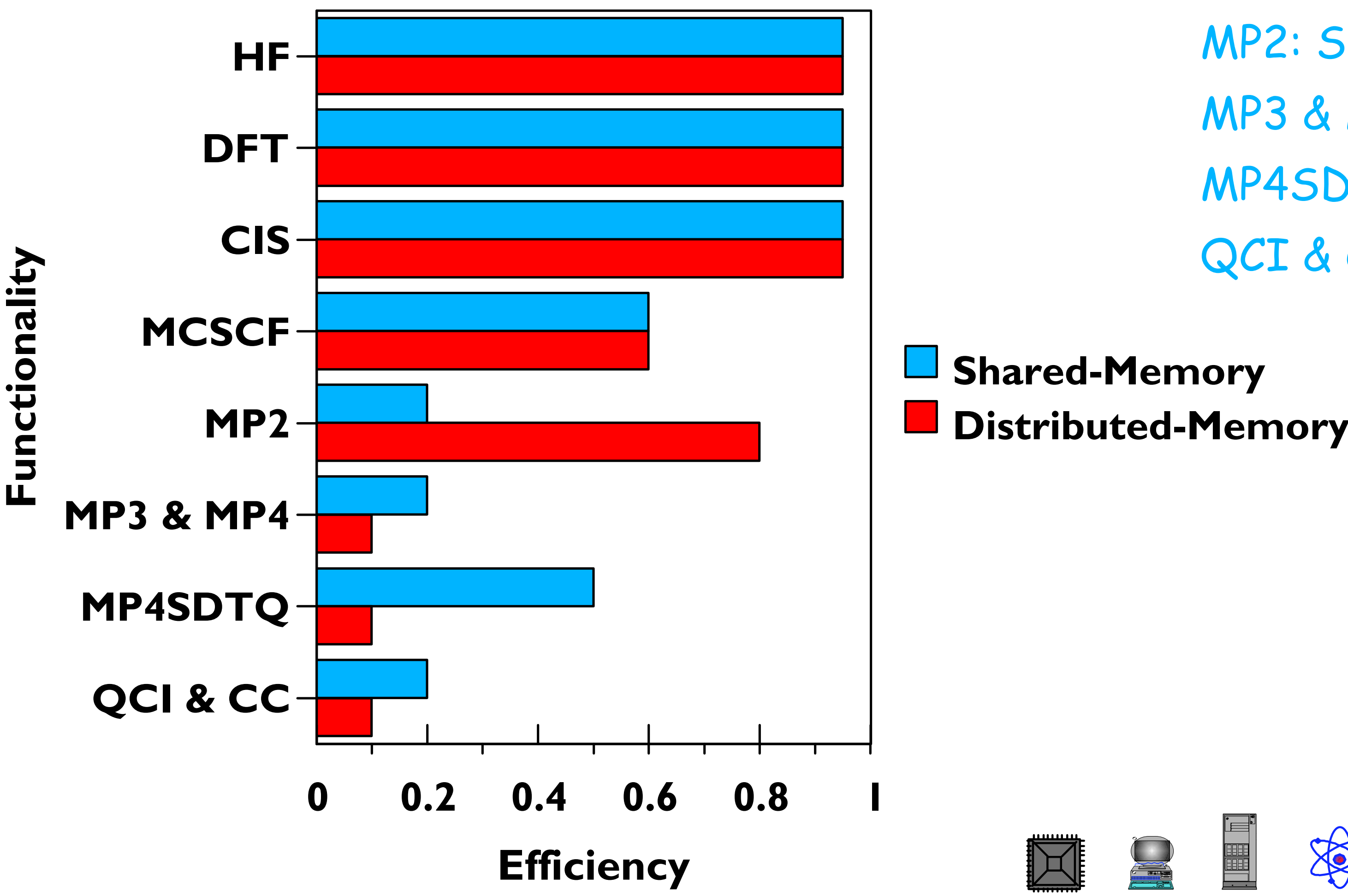
MCSCF: SP and Gradients

MP2: SP, Gradients

MP3 & MP4: SP

MP4SDTQ: SP

QCI & CC: SP



Σ

Information

Gaussian official site:

http://www.gaussian.com

Institute-IBM Gaussian site:

http://www.msi.umn.edu/user_support/compchem/gaussian_tech/

contact:

help@msi.umn.edu