An overview of the R-matrix electron-impact scattering package at NERSC

Workshop on Leadership-class Machines, Petascale Applications and Performance Strategies

July 27-30, Granlibakken Resort

Connor Ballance

Auburn University

Collaborators: D C Griffin
: S D Loch
: N R Badnell
: M S Pindzola

Acknowledgments: Dept of Energy
: NERSC
Structure of Talk

1. What is the R-matrix method?

2. What is the underlying computational formalism and structure of the code base?

3. Scaling up from 1000 -> 10000 processors on franklin.

4. Current bottlenecks/possible future directions
\[ R_{ij} = \sum_k \frac{w_{ik} w_{jk}}{E_k - E} \]
Computational Method

1. The formation of many real symmetric matrices (Hamiltonians), typically 60-100, requiring anywhere from 10-500 Gb of storage.

2. The diagonalisation of each matrix, from which EVERY eigenvalue and EVERY eigenvector is required.

We achieve this through the ScaLapack package and in particular routines: *pdsyevx* and *pdsyevd*. Preference is given to the latter as it ensures orthogonality between all eigenvectors.

Matrices vary in size from $2K \times 2K$ to $100K \times 100K$, depending on the complexity of the atomic target.
LS coupling R-matrix (exchange) flowchart

INPUT

das/dauto

AUTOSTRUCTURE

OUTPUT

radial


dstg1

pstg1r

STG1 DAT
RXXXX.DAT


dstg2

pstg2r

SIG2XXXX.DAT
scaled.dat


dstg3

pstg3r

H.DAT

Outer Region


dstgf

pstgf

OMEGA
pstgf is well tuned ... DGEMM and loop-unrolling, make future impact on code optimisation difficult!

FRANKLIN (2500 procs) during the dual core period
The formation of the scattering Hamiltonian

The formation of the continuum-continuum part of the N+1 electron Hamiltonian is the most time consuming ....

<table>
<thead>
<tr>
<th>Number of channels</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Matrix Blocks" /></td>
</tr>
</tbody>
</table>

Therefore, if there are several thousand scattering channels

\[
\frac{(NCHAN\times(NCHAN+1))}{2}
\]

matrix blocks !!!

i.e. several million
not very subtle, but effective .... parallel concurrent diagonalisation

( with help from NERSC consultants: Woo-Sun Yang, Katie Antypas )

\[
\begin{array}{cccccccccc}
\text{pstg3r\_split} & \text{ie 5200 processors} \\
\text{mpi\_comm\_split} & 1 & \text{...} & N
\end{array}
\]

Each small box represents a partial wave

Each subgroup reads a single Hamiltonian and diagonalises it in parallel, concurrently with every other.

Almost endlessly scalable

Reduces all future R-matrix close-coupling calculations to the time required for a single partial wave.
Present situation when 'scaling up' on franklin .....  

96 matrices, each 10721*10721 in dimension
sample profiling for the same case ...

```
Totals for program

<table>
<thead>
<tr>
<th>Time%</th>
<th>100.0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>179.868711 secs</td>
</tr>
<tr>
<td>Imb.Time</td>
<td>0.485217 secs</td>
</tr>
<tr>
<td>Imb.Time%</td>
<td>0.3%</td>
</tr>
<tr>
<td>Calls</td>
<td>2.6 /sec</td>
</tr>
<tr>
<td></td>
<td>405.0 calls</td>
</tr>
</tbody>
</table>

RETIRED_MMX_AND_FP_INSTRUCTIONS:

<table>
<thead>
<tr>
<th>PACKED_SSE_AND_SSE2</th>
<th>1722.618M/sec</th>
<th>272399534900 instr</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_FML_INS</td>
<td>1076.992M/sec</td>
<td>170305883797 ops</td>
</tr>
<tr>
<td>PAPI_PAD_INS</td>
<td>1081.897M/sec</td>
<td>171081650827 ops</td>
</tr>
<tr>
<td>PAPI_FDV_INS</td>
<td>2.185M/sec</td>
<td>345548107 ops</td>
</tr>
<tr>
<td>User time (approx)</td>
<td>158.131 secs</td>
<td>363701551563 cycles 87.9%Time</td>
</tr>
<tr>
<td>Average Time per Call</td>
<td>0.444120 sec</td>
<td></td>
</tr>
<tr>
<td>CrayPat Overhead : Time</td>
<td>0.0%</td>
<td></td>
</tr>
<tr>
<td>HW FP Ops / Cycles</td>
<td>0.94 ops/cycle</td>
<td></td>
</tr>
<tr>
<td>HW FP Ops / User time</td>
<td>2158.889M/sec</td>
<td>341387534625 ops 23.5%peak(DP)</td>
</tr>
<tr>
<td>HW FP Ops / WCT</td>
<td>1897.982M/sec</td>
<td></td>
</tr>
<tr>
<td>FP Multiply / FP Ops</td>
<td>49.9%</td>
<td></td>
</tr>
<tr>
<td>FP Add / FP Ops</td>
<td>50.1%</td>
<td></td>
</tr>
<tr>
<td>MFLOPS (aggregate)</td>
<td>3316053.72M/sec</td>
<td></td>
</tr>
</tbody>
</table>
```

---

The profiling data shows the time and operations performed during the execution of the program. The total time is 179.868711 seconds, with 0.485217 seconds spent on imbalanced time, representing 0.3% of the total time. The program made 405.0 calls. The retired MMX and FP instructions are detailed, including the number of instructions and operations performed. The average time per call is 0.444120 seconds, and the CrayPat overhead is 0.0%. The hardware floating-point operations per cycle and per user time are also provided, along with the MFLOPS (millions of floating-point operations per second) aggregate.
In a nutshell, one of the biggest remaining challenges to continued scaling of the problem is:

the construction of the Hamiltonian,

then distributing the pertinent information to

a particular processor,

in a particular ScaLapack context,

before diagonalisation.