Code optimization for Cell/B.E.
Opportunities for ABINIT – a software package for physicists

ABINIT on Cell - Overview
- The Cell/B.E. processor (aka “Cell”) developed by Sony, Toshiba and IBM is a heterogeneous microcomputer processor.
- This architecture offers a great peak performance for scientific computations.
- We took some opportunities to optimize ABINIT for Cell and present first results.

ABINIT:
- A software package to compute the total energy, charge density and electronic structure of systems made of electrons and nuclei.
- 240.000 lines of Fortran code.
- Uses MPI for parallelization.

Profiling ABINIT promised that optimizing a few functions should lead to a serious speedup of the whole application, in fact 4765 (2%) source lines of code (SLOC) make up 87% of ABINIT runtime.

<table>
<thead>
<tr>
<th>Function</th>
<th>Runtime</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZGEMM</td>
<td>25%</td>
<td>SLOC</td>
</tr>
<tr>
<td>opdotkk</td>
<td>35%</td>
<td></td>
</tr>
<tr>
<td>fftsp</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>mktsp3</td>
<td>7%</td>
<td></td>
</tr>
<tr>
<td>pw_python</td>
<td>5%</td>
<td></td>
</tr>
</tbody>
</table>

We started by optimizing ZGEMM because the operation which is done by this routine can be understood quite easily, so we could focus on optimization and getting familiar with the Cell programming environment.

The complex multiplication described above, implemented in C:

```
for (k=0; k < kmax; k++)
    for (kk=0; kk < kmax; kk++)
        for (i=0; i < nlocal; i++)
            for (j=0; j < nlocal; j++)
                a[i][j] = re(i)*re(j) + im(i)*im(j);
```

Our ZGEMM implementation is 40 times faster for 2000x2000 square matrices than the ZGEMM implementation in the refill package.

Math kernel optimization
Our BLAS3/ZGEMM implementation:
- Parallel multiplication of complex matrices with double precision.
- Whole computation is done on the SPEs, PPE only administers SPE threads.

The unmodified version of ABINIT is roughly twice as fast on a 2 GHz Opteron as on a Cell. If we manage to optimize the other compute kernels by the same factor as ZGEMM the Cell version could be more than three times faster on the Cell than on the Opteron.

To simplify the process of porting math kernels to the Cell platform we are currently about to build tools which help with optimizing the compiler generated (gcc -O3) assembly, similar to spu_timing but in a more ‘active’ way, which means that the pipeline status should not only be viewable but optimizations should be suggested.

```
The current SDK does not offer a ZGEMM implementation, thus we used DGEMM for comparison.
```

The Cell SDK 3.0 (pre-release) achieves 9.5 GFlop/s ZGEMM performance for a 2000x2000 matrix. This corresponds to 68% of the Cell's peak performance. Our optimized ZGEMM implementation is able to leverage up to 73.5% of the peak performance even though the complex multiplication requires more shuffle operations.

```
The unmodified version of ABINIT is roughly twice as fast on a 2 GHz Opteron as on a Cell. If we manage to optimize the other compute kernels by the same factor as ZGEMM the Cell version could be more than three times faster on the Cell than on the Opteron.
```

Future Work
- Optimization of the other ABINIT compute kernels for Cell
- Exploring ways to efficiently use heterogeneous clusters for ABINIT

This research is supported by the Center for Advanced Studies (CAS) of the IBM Böblingen Laboratory as part of the NICOLL Project.