

# Scaling behavior of IMPACT+mxyzptk on various architectures

James Amundson

21st October 2002

## 1 Introduction

This document describes my preliminary attempt to understand the performance of IMPACT+mxyzptk for FNAL Booster studies. IMPACT is a Fortran 90 accelerator simulation code with a parallelized 6D space charge implementation. mxyzptk is a C++ library that we use to provide maps for the Booster. This study is preliminary; I have made no serious attempt to verify the reproducibility of the results.

## 2 The machines

### 2.1 Seaborg

Seaborg is NERSC's IBM SP RS/6000 with 2,944 375 MHz POWER3 processors. For details see <<http://hpcf.nersc.gov/computers/SP/>>. We use the IBM Fortran compiler and g++ on Seaborg.

### 2.2 Alvarez

Alvarez is NERSC's Linux cluster of 80 dual 866 MHz PIII processors connected with both Ethernet and Myrinet networking. I used the Myrinet for these tests. We use the Portland Group Fortran compiler and g++ with the MPICH MPI libraries on Alvarez.

### 2.3 Heimdall

Haimdall is the FNAL Beams Theory Group's Linux cluster of 32 dual Athlon 1.4 GHz processors connected with 100 Mbit/s Ethernet. Heimdall also has Gigabit Ethernet, but it is waiting for a backordered switch. We use the Portland Group Fortran compiler and g++ with the LAM MPI libraries on Heimdall. We have also run tests using the Intel Fortran Compiler on Heimdall. In both

cases, the code was compiled with “-O2 -g”. The performance using the Portland Group Compiler can be improved through a more judicious set of flags, however in no case was I able to get close to the performance I saw using the Intel Compiler. Compiling under Intel with “-O3” produced a small, negative effect.

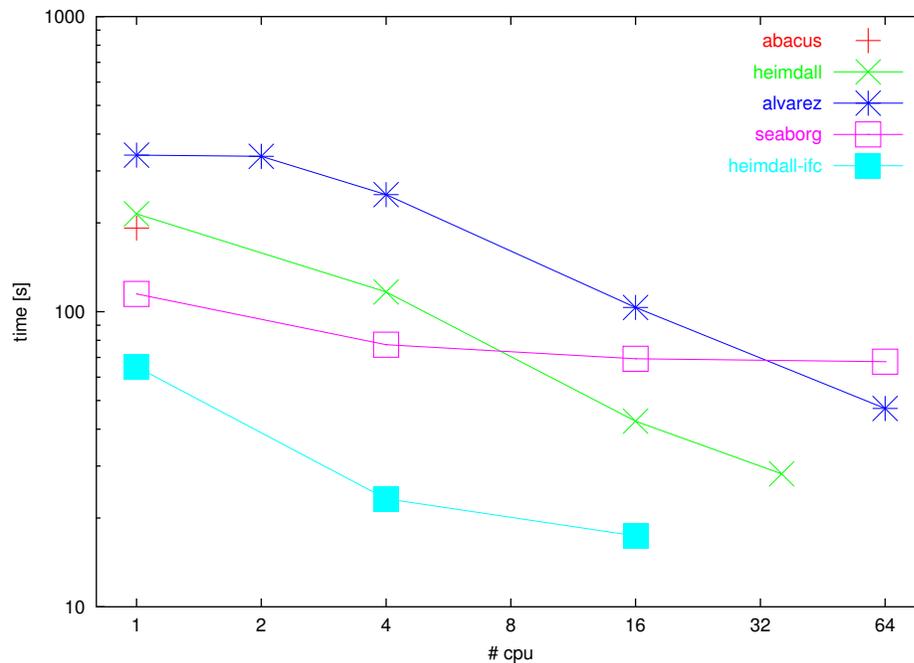
## 2.4 Abacus

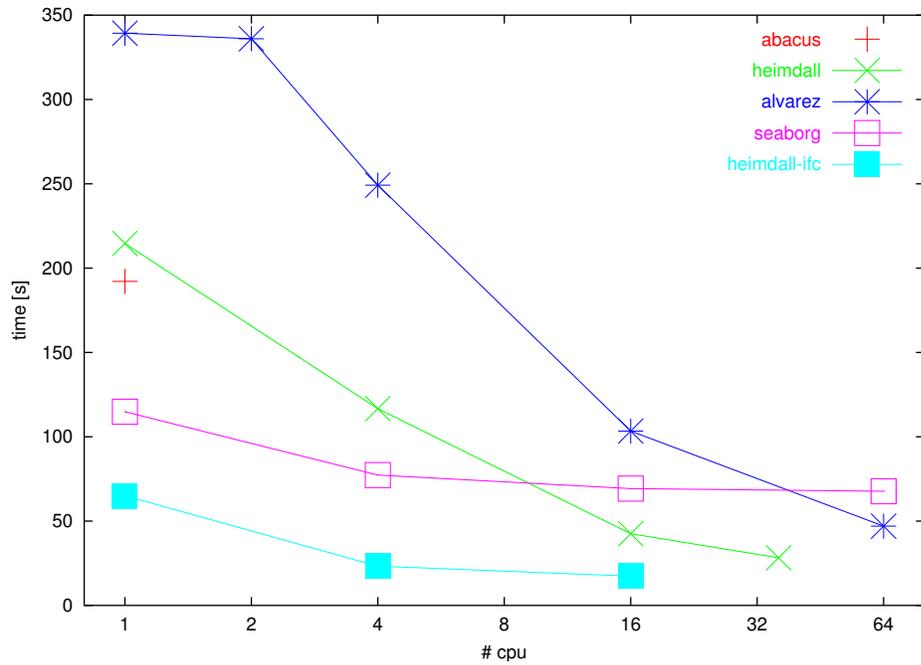
Abacus is my Laptop with (single!) 800 MHz PIII processor. I use the Intel Fortran compiler and g++ and the MPICH MPI libraries on my laptop.

# 3 The tests

## 3.1 No space charge

For the first test, I ran 100K particles through a single turn of the Booster without space charge effects. The wrapper code with all the input parameters is stored in the Impact3d cvs module in the file booster\_test1.py. I told impact to use a symmetric geometry for all multi-cpu runs, e.g 64 cpus ran in the 8x8 configuration. I am presenting the data with both linear and logarithmic time scales. The latter is better for displaying scaling behavior, while the former is better at indicating perceived speed.





### 3.2 Space Charge

The second test involved IMPACT's space charge routines. I used 2.7M particles on a 65x65x65 grid. The particles travelled through one turn of the booster with 100 space charge kicks. The wrapper code with all the input parameters is stored in the Impact3d cvs module in the file booster\_test2.py. I used the symmetric geometry for all but one run; the faster 36 processor run on heimdall used the 4x9 configuration.

