# Technical Report TR08

Study of SpMV performance using PETSc

Vennila Megavannan, Omkar Deshmukh,
Naveen Anand Subramaniam and Dan Negrut

July 6, 2015

# Contents

# 1 Introduction

This report evaluates the performance of SpMV using the PETSc library on Intel and AMD architectures and on multiple nodes. PETSc (Portable, Extensible Toolkit for Scientific Computation) provides object oriented Matrices and Vectors which are internally represented as MPI. SpMV suffers from higher instruction and storage overheads per flop, as well as indirect and irregular memory access patterns. Hence SpMV is memory bound and several optimizations are required such as low level code optimizations, data structure optimizations and parallel optimizations.

# 2 Related Work

Williams et. al have done an extensive study of sparse matrix-vector multiply (SpMV) performance on hardware platforms such as AMD and Intel quad-core architectures, STI Cell, Sun UltraSPARC Victoria Falls. The authors have implemented various SPMV optimizations such as thread blocking, TLB blocking, Register blocking, NUMA aware memory allocation and thread affinity, software prefetching and the performance of the optimized implementation is compared against off the shelf OKSI-PETSC performance. OKSI-Petsc is referred to the MPI-based distributed memory SpMV in PETSc 2.3.0 with the serial SpMV component replaced with calls to low-level primitives that provide auto-tuned computational kernels on sparse matrices. The best performance obtained on AMD Opteron was 2 to 3.3 GFLOP/s . Matrices with low flop:byte ratios show poor parallelization and cache behavior achieving performance of only 0.5 to 1 Gflop/s. The OKSI-PETSC performance ranges from 0.5 to 2 GFlop/s. According to the authors, OSKI-PETSc yields only moderate speedups as communication time accounts to 30 % of the total SpMV execution time and explicit memory copies accounts to 56% of the execution time.

# 3 Hardware Platforms Used

Following are the processors in which we ran our experiments.

## 3.1 Intel Xeon E5 - 2690 v2

**MicroArchitecture**: Ivy Bridge
**Level 1 cache size** : 10 x 32 KB 8-way set associative instruction caches, 10 x 32 KB 8-way set associative data caches
**Level 2 cache size**: 10 x 256 KB 8-way set associative caches
**Level 3 cache size**: 25 MB 20-way set associative shared cache
**Memory**: 64GB
**Quick Path Interconnect** (2 links)
**Number of Cores**: 10

**Number of Threads** : 20
**Maximum memory bandwidth (GB/s)**: 59.7
**Memory channels**: 4
**DIMMs per channel**: 3
**No. of Sockets**: 2

### 3.2 AMD Opteron 6274

**MicroArchitecture**: Bulldozer
**Level 1 cache size**: 8x64 KB 2-way Instruction 16x16KB 4-way Data
**Level 2 cache size**: 8x2 MB 16-way set associative caches
**Level 3 cache size**: 2x8 MB up to 64-way set associative shared cache
**Memory**: 64GB
**HyperTransport 3.0 technology** (4 HT links)
**Number of Cores**: 16
**Number of Threads** : 32
**Maximum memory bandwidth (GB/s)**: 51.2
**Memory channels**: 4
**DIMMs per channel**: 3
**No. of Sockets**: 4

## 4 Experiments

We used the matrix multiplication routine (MatMult) defined by PETSc. The matrices (MatMult) are input in binary format. One of the optimizations that PETSc recommends to hide the latency caused by page fault is to first execute a smaller system before executing the actual matrix. We have implemented this in our code. SpMV is run with the input matrices described in 4.1 on hardware platforms described in 3 with 16 processors and 32 processors and on multiple nodes.

### 4.1 Input Matrices

Actual matrices from a simulation were used to gauge real world performance. The matrices have row size of 3.13 million and column size ranges from 3.2 to 3.6 million. The number of non zeros range from 19 million to 21.6 million. Six different matrices are compared on each platform.

# 5 Results

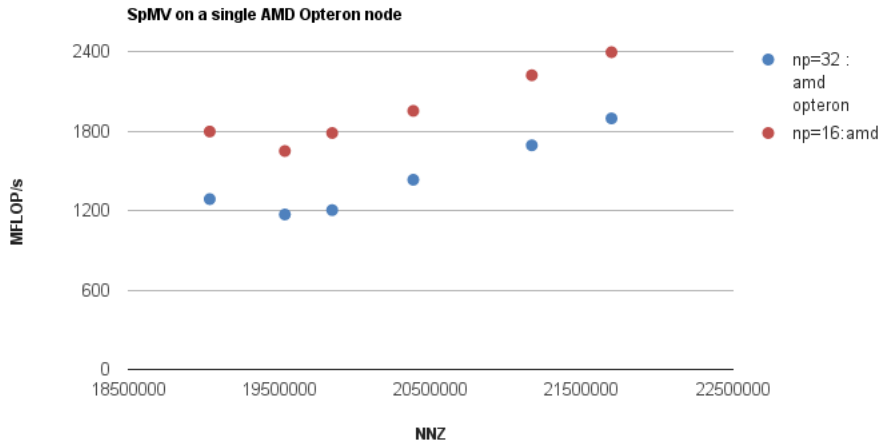


Figure 1: Performance of SpMV on a AMD Opteron Node

Figure 1 compares the performance of SpMV when it runs a single AMD Opteron node with 16 processors vs when it runs with 32 processors. NNZ (number of non zeros) on the X-axis represents the number of non zeros in the matrix. It is observed that 16 processors performs better than 32 processors.

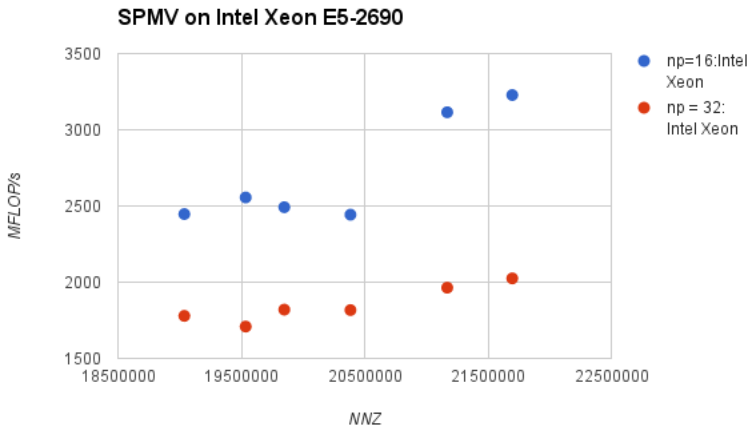


Figure 2: Performance of SpMV on a AMD Opteron Node

Figure 2 compares the performance of SpMV when it runs a single Intel Xeon node with 16 processors vs when it runs with 32 processors. It is observed that 16 processors performs better than 32 processors in this case as well.

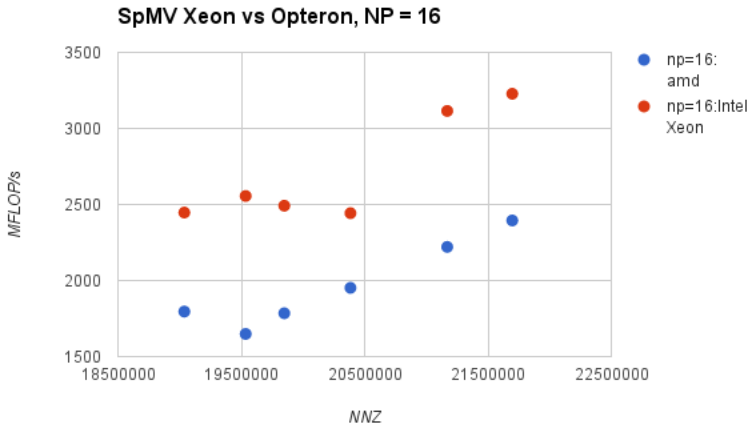


Figure 3: Comparison of SpMV on AMD Opteron vs Intel Xeon with 16 processors

Figure 3 compares the performance of SpMV on AMD Opteron vs Intel Xeon with 16 processors. It is observed that the Intel Xeon E5 - 2690 v2 processor performs better than the AMD Opteron processor.

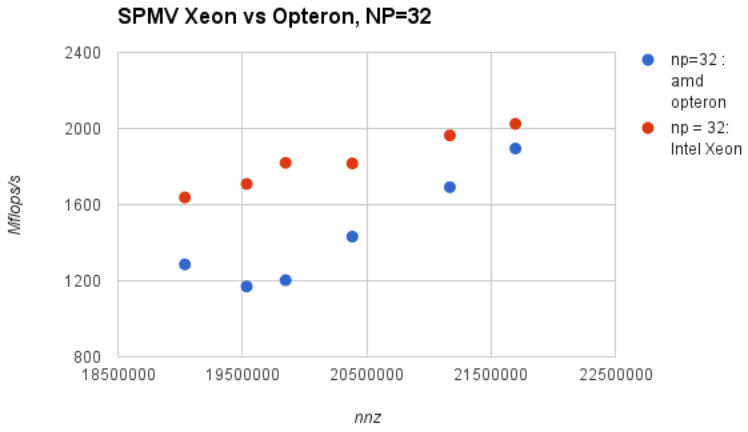


Figure 4: Comparison of SpMV on AMD Opteron vs Intel Xeon with 32 processors

Figure 4 compares the performance of SpMV on AMD Opteron vs Intel Xeon with 32 processors. It is observed that the Intel Xeon E5 - 2690 v2 processor performs better than the AMD Opteron processor in this case as well.

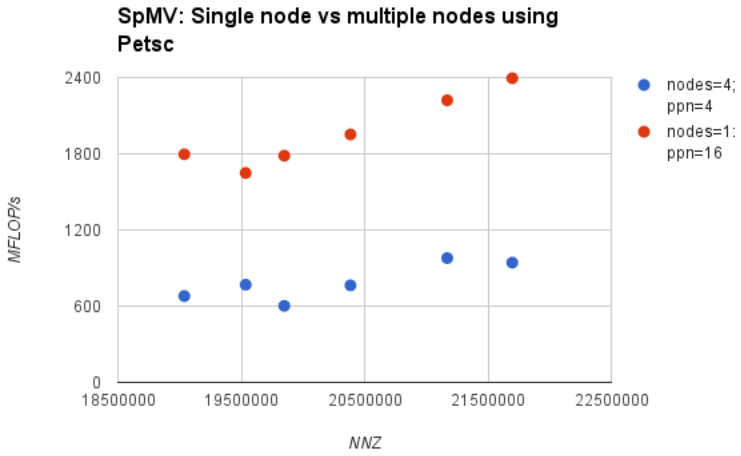


Figure 5: Comparison of SpMV on multiple nodes vs single node

Figure 5 compares the performance of SpMV on multiple nodes vs single node with 16 processors . It is observed that performance is better on a single node with same number of processors than on multiple nodes.

# 6 Inferences

We observed that SpMV performs better on 16 processors as compared to 32 processors. This is probably because the sparse matrix vector multiplication required more memory bandwidth and interconnects bandwidth as compared to compute capacity and running it on 32 processors lead to more communication time hence it did not perform as well as the 16 processors. Also, the performance mostly improved with increase in NNZ. It also depends on the number of rows and columns. When ran on multiple nodes, the performance decreased more drastically as the GFOLP/s is limited by the communication bandwidth. Also it was observed that Intel Xeon performs consistently better by 1 GFLOP/s as compared to AMD opteron.