Auto-Tuning OmpSs-OpenCL Kernels across GPU Machines

Vinoth Krishnan Elangovan, Rosa. M. Badia and Eduard Ayguade
Barcelona Supercomputing Center and Universitat Politécnica de Catalunya

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Auto-Tune - Motivation

- GPUs have become focal point in todays HPC.
- GPUs architectural upgrades are quite predominant with every generation release.
- With every release, Performance Portability across these machines is desired.
- Lack of tools hinder this smooth portability, hence burdening the programmer.
- We address this issue with an Auto-Tune Tool focusing on modifying OpenCL kernel execution configuration based on GPU characteristics.

OmpSs-OpenCL Programming Model

- OpenCL support for OmpSs model using annotations for parallel regions(Directive based).
- Task based model with an asynchronous execution model following a sequential style of programming supporting Heterogeneous devices(both OpenCL CPU and GPU).
- Comprises of Mercurium compiler and Nanos runtime which maintains data dependency and implements the asynchronous execution of tasks.

OmpSs-OpenCL Example

```
1. #pragma omp target device (clcpu) ndrange(1,size,512) copy_deps
2. #pragma omp task input ([size]x) output ([size]y)
3. void copy_task(double *x,double *y,int size);
4. #pragma omp target device (clgpu) ndrange(1,size,256) copy_deps
5. #pragma omp task input ([size]b, [size]c) output ([size]a)
6. void triad_task (double *a, double *b, double *c, double scalar, int size);
7. int main(int argc, char** argv)
8. copy_task (x,y,size);
9. triad_task (a,b,c,scalar,size);
10. #pragma omp taskwait
```

OmpSs-OpenCL Kernel/Task

- __kernel void triad_task (__global double *a, __global double *b, __global double *c, __const double scalar, __const int size)
- int j=get_global_id(0);
- a[j] = b[j]+scalar*c[j];

Auto-Tune Model

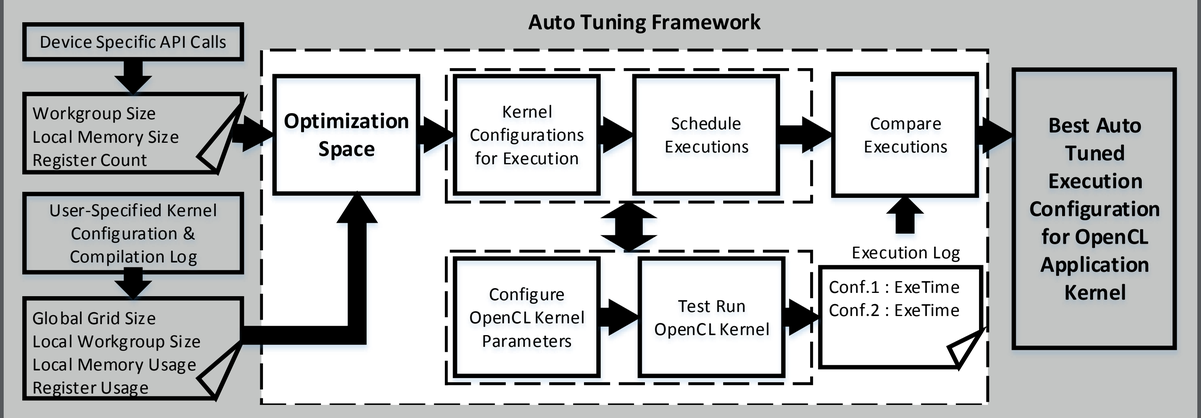

Figure 1: Auto-Tune Model

Key-features Tuned

- Work-Group sizes are tuned in such a manner that the kernel fits the local Memory, Register Count and Hardware Threads specific to the GPU.
- An optimization Space is created consisting of all possible kernel configuration for a single GPU.
- The set is executed to find the best mapping of the threads (work-group size) onto the GPU cores.
- The kernel characteristics and the device features are obtained using OpenCL API calls and kernel compilation log.
- This Auto-Tune Tool fits perfectly for OmpSs-OpenCL Task-based programming model.

Analysis

- We investigated the tool with 3 different benchmarks with 3 different GPUs.
- We focused on Matmul and histogram using local memory and register-based matmul benchmarks for our analysis.

GPU	SMs	Local Memory	NDRange	Registers	SM-Threads
Tesla M2090(GPU 1)	16	49152	1024	32k	1536
NVS 3100M(GPU 2)	2	16384	512	16k	1024
GT 630M(GPU 3)	2	49152	1024	32k	1536

Results-Matmul

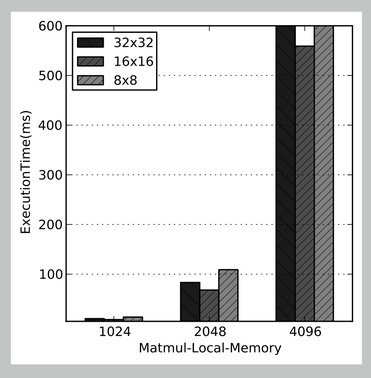

Figure 2: Tesla M2090

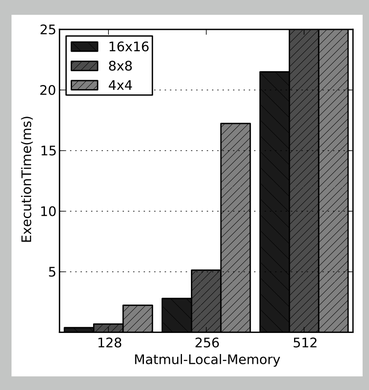

Figure 3: NVS 3100M

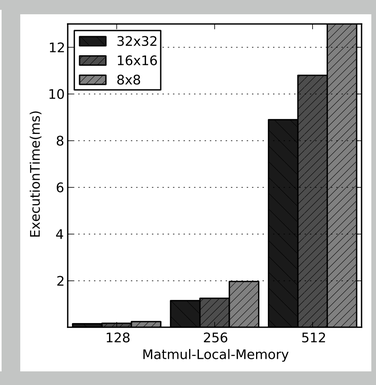

Figure 4: GT 630M

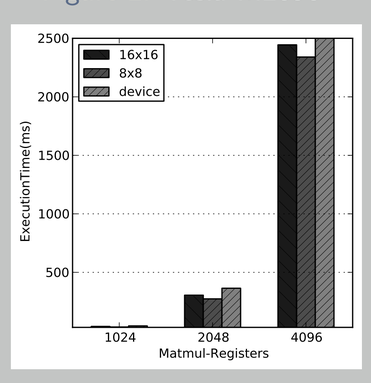

Figure 5: Tesla M2090

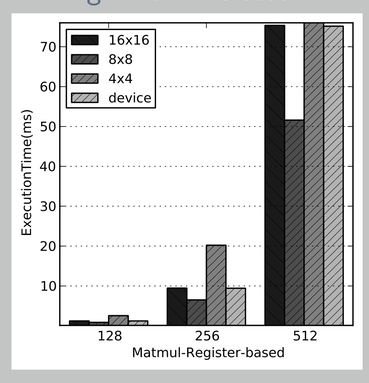

Figure 6: NVS 3100M

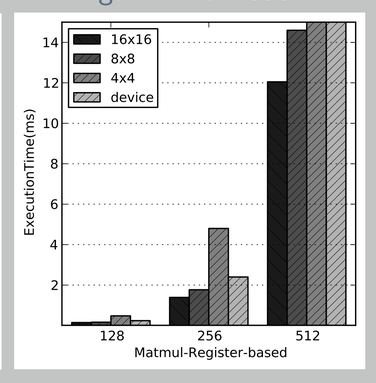

Figure 7: GT 630M

Results-Histogram

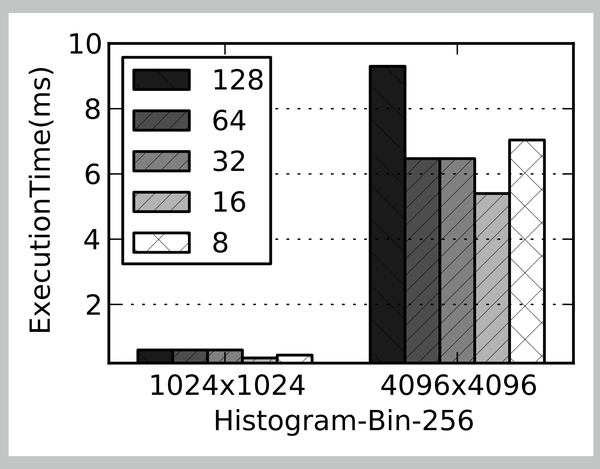

Figure 8: Tesla M2090

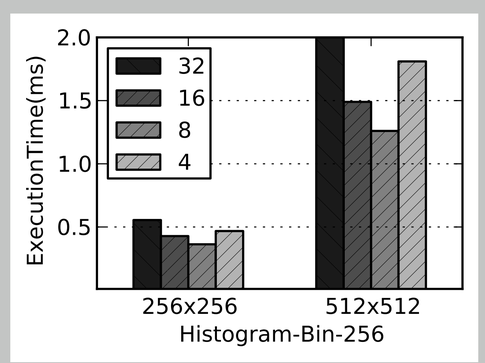

Figure 9: NVS 3100M

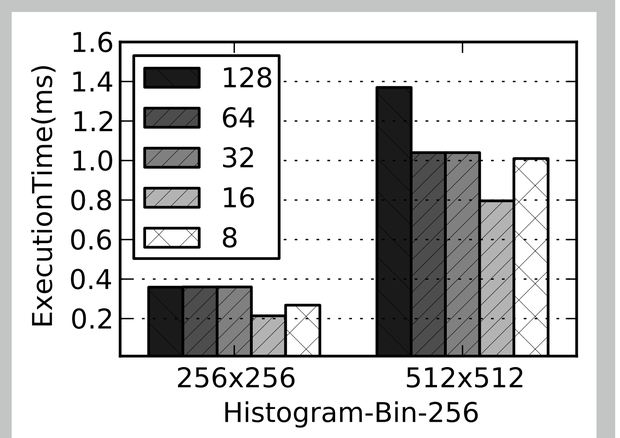

Figure 10: GT 630M

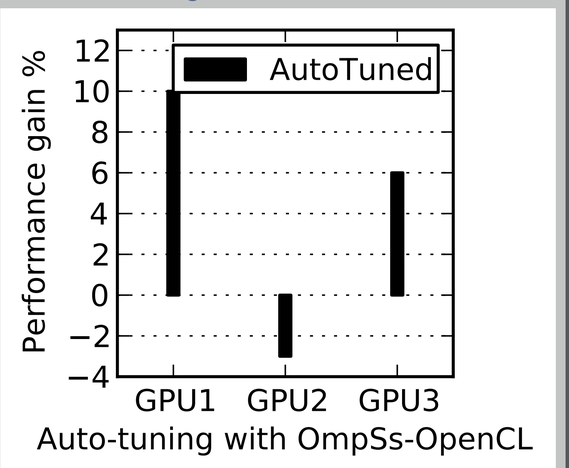

Figure 11: OmpSs-OpenCL

Conclusion

- Auto-tune Tool addresses one aspect in OpenCL performance portability across GPU machines.
- The investigation with OmpSs-OpenCL delivered an average performance increase of 5% for 64 blocked matmul tasks across 3 GPUs.
- In GPU1 & GPU3 user configuration fails to give the best performance and tuning helps in providing 10% & 6% gain. Whereas in GPU2, when user specification is optimal and tuning becomes an overhead of 3%.

vinoth.elangovan@bsc.es