Efficiency considerations in heterogeneous cluster systems

Valon Raca and Eduard Mehofer

Faculty of Computer Science
University of Vienna, Austria
{valon.raca,eduard.mehofer}@univie.ac.at

Abstract. The importance of heterogeneous asymmetric clusters has grown steadily over the last years. Such architectures pose new challenges with respect to execution time and energy consumption. Work distribution for homogeneous systems has been done typically by dividing the work by the number of compute nodes and assigning equal-sized portions to each of them. Such an approach is not adequate any more for heterogeneous systems. The amount of work has to be adjusted to the computational power of the individual devices. Moreover, heterogeneity of devices raises in addition to execution time a second issue – the energy consumed to fulfill a task. A work distribution approach could e.g. exclude low-performing, high-energy-consuming devices from execution. Optimization in one direction only, i.e. execution time or energy consumption, does not meet the requirements. In this paper we discuss different work distribution approaches and report our experiences.

Keywords: heterogeneous clusters, runtime efficiency, energy efficiency

1 Introduction

The trend of using accelerators for solving time-consuming problems is continuing to increase steadily. Devices such as GPUs, Intel Xeon PHI, or FPGAs are performing much better than CPUs for many data-parallel classes of applications, thus becoming a mainstream architecture in HPC. However, heterogeneous systems require adequate support for handling programming obstacles resulting from the variety of hardware characteristics of those devices. The emergence of OpenCL has alleviated very much the burden of programmers dealing with different types of devices. OpenCL provides a platform for abstracting the hardware disparities through its transparent model which assists programmers writing codes which can be executed on any OpenCL enabled device.

Large clusters consisting of heterogeneous computing nodes with different computing devices provide the opportunity to scale-up the performance. OpenCL can adapt codes for a large number of devices from different vendors, but it does not support data transfers and coordination tasks within a cluster. In addition libraries like MPI are required. Further, the arrangement of accelerators in cluster compute nodes needs not to be symmetric. The devices can be distributed

unevenly between the nodes resulting in an asymmetric configuration of a cluster.

Work distribution for such heterogeneous, asymmetric clusters becomes a major problem. Whereas for homogeneous clusters usually equal-sized portions of work has been distributed to the compute nodes, this approach is not adequate for heterogeneous systems any more. The amount of work has to be adjusted to the computational power of the individual devices. Moreover, energy-efficiency is another issue introduced by heterogeneous systems which is gaining increasing attention. A work distribution approach could e.g. exclude low-performing, high-energy-consuming devices from execution. Optimization in one direction only, i.e. execution time or energy consumption, does not meet the requirements. Energy-efficiency together with performance has to be taken into account in the time-energy problem space when dealing with efficiency issues.

We consider applications with a workload that is partitioned into work packages which are assigned to devices to be processed. Our programming and hardware model is described in detail in [12]. Runtime efficiency and energy efficiency basically boils down to the problem of mapping work packages to devices. Different work distributing strategies are possible resulting in different solutions exhibiting different efficiency characteristics. An assessment of a solution heavily depends on the requirements of the programmer. When we optimize in two directions, the mapping of work packages to devices results in a solution space containing 4 distinguished solutions exhibiting optimality criteria:

- *best performance*: one dimensional problem–energy consumption neglected
- *minimal energy*: one dimensional problem–execution time neglected
- *minimal energy with restrictions to performance*: two dimensional problem
- *best performance with restrictions to energy*: two dimensional problem

In addition to the 4 solutions above, trade-off solutions between time and energy can be considered as well. This papers discusses all the different solutions in detail and presents the impacts in practice.

The rest of the paper is organized as follows. Section 2 outlines our programming approach for heterogeneous, asymmetric clusters. Section 3 gives the motivation for efficiency considerations in heterogeneous asymmetric clusters. Optimization problems are discussed in Section 4. We survey the related work in Section 5 and conclude with a summary in Section 6.

2 Programming Support for Heterogeneous Asymmetric Clusters

In this section we give a brief overview of our framework [12] which supports heterogeneous, asymmetric cluster architectures. The goal of the framework is to assist a programmer to run an application efficiently in such an heterogeneous environment. Our strategy to deal with such clusters is to partition the work into work packages which are assigned to compute devices to be processed. As shown

in Fig. 1, the input is partitioned in smaller data chunks called *work packages WP1,...,WPn* which are equal-sized. The size of work packages is determined by taking various hardware and application characteristics into account like memory capacity or peak performance.

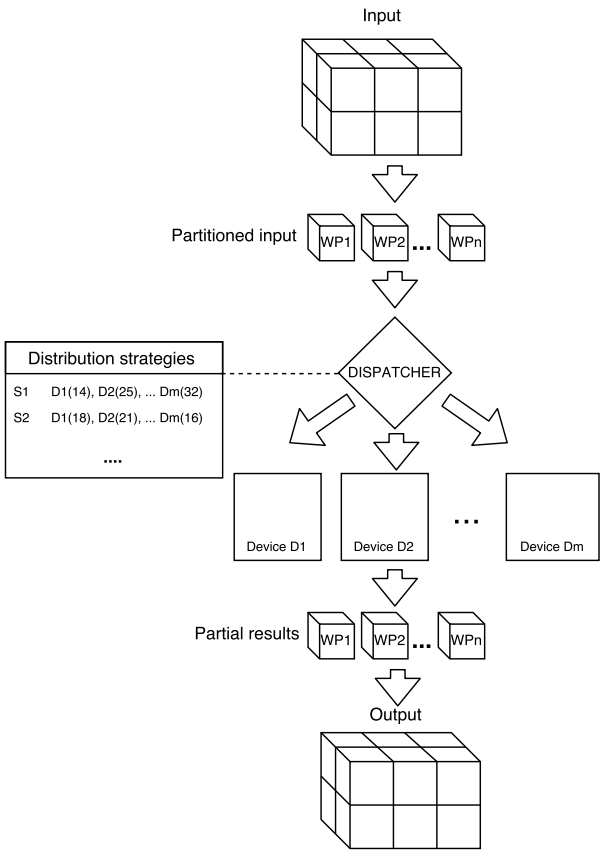


Fig. 1: Main application processing steps

Usually the number of work packages are orders of magnitude greater than the number of available devices in the system. This problem coupled with different performance and energy consumption numbers for each of the devices leads to the requirement for a strategy in distributing work packages. The different types of mappings is realized by a *dispatcher* which manages the distribution of work packages. As shown in Fig. 1, the dispatcher can perform different distribution strategies $(S1, S2, ...)$ which define different mappings; e.g. strategy $S1$ requires that 14 WPs are assigned to device $D1$, 25 WPs to $D2$, and so forth, whereas strategy $S2$ requires that 18 WPs are assigned to device $D1$ and 21

WPs to $D2$. The output is constructed from the partial results of the processed work packages. The framework helps the programmer to run applications in such heterogeneous environments without dealing with hardware configurations or communication issues explicitly.

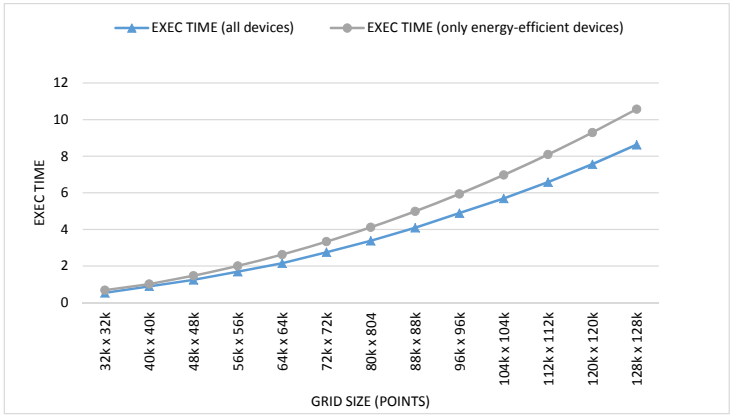
3 Examples for Selecting Devices and Distributing Work

As discussed above, work distribution is done by mapping work packages onto compute devices. Hence, different mappings result in different execution times and different power draws. Depending on whether execution time or energy consumption is favored by a programmer, some mappings fit better to the needs than others. In the following we will discuss three examples which have the common property that better performance is achieved at the cost of higher energy consumption. Or in other words: longer execution times may help to save energy. These examples show that getting better in one dimension may degrade the other dimension which means that an optimization has to take both dimensions into account.

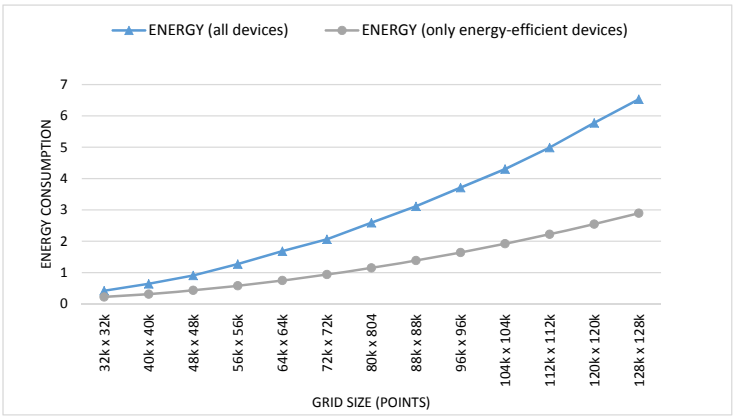
The first example is taken from a paper by Liu and Luk [11]. As shown there, the FPGA is the most energy-efficient device for executing function SGEMM (matrix-matrix operation $C = \alpha AB + \beta C$) followed by GPU and CPU. Using only the FPGA for processing SGEMM leads to the smallest amount of energy consumption, but it may take about 40 times longer to complete than using the GPU.

We have done a similar experiment on our PHIA cluster which consists of 8 compute nodes with devices such as NVIDIA GPUs and Intel XEON Phi accelerators arranged in an asymmetric configuration. Our experimental results confirm that execution time and energy consumption can be influenced to a great extent based on the devices used in computing. In Fig. 2 we show the execution time and energy consumption in normalized units for a molecular analysis kernel with different problem sizes. As it is shown in Fig. 2a when we use only most energy-efficient devices the execution time (circular markers) increases in comparison to the program version where we use all available devices of the system (triangular markers). The gap between the two execution scenarios increases steadily with bigger problem sizes. However, in Fig. 2b it can be seen that energy consumption is reduced significantly when only the most energy-efficient devices are used in computation (circular markers). The figures show that for all problem sizes a loss in performance is rewarded by a reduction of energy consumption.

In addition to selecting devices for execution, the next example shows that the distribution of work onto the selected devices plays an important role as well. To demonstrate this, let us assume we have 5 equally-sized work packages and two devices with the following time and energy values per work package $T = \{2, 5\}$ and $E = \{10, 5\}$, i.e. 2 and 5 time units are taken for a work package on device 1 and 2, and 10 and 5 energy units on the devices, respectively. Further a time constraint is specified which requires that execution should finish within



(a) Execution time for all devices used vs. energy-efficient devices only



(b) Energy consumption for all devices used vs. energy-efficient devices only

Fig. 2: Performance and energy figures in our PHIA cluster for DCS

10 time units. The following mappings of work packages onto devices are feasible and fulfill the time constraint:

- If we assign 4 work packages to the first device and 1 work package to the second device, the overall execution time equals to $max\{8, 5\} = 8$ units of time, while the overall energy consumption equals to $10 * 4 + 5 = 45$ units of energy consumption.
- If we assign 3 work packages to the first device and 2 work packages to the second device, the overall execution time equals to $max\{6, 10\} = 10$ units of time, while the energy consumption equals $10 * 3 + 5 * 2 = 40$ units.

Both work distributions satisfy the time constraint, however the second work distribution is more energy-efficient than the first one.

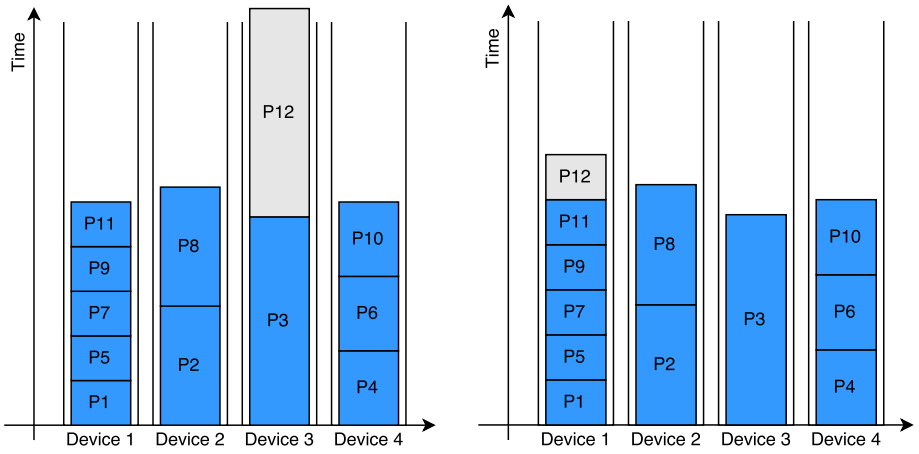
4 Different Solutions Satisfying Optimality Criteria

Best performance. In principle the best performance approach leads to a solution where all devices are used for computation. However, there are situations where this might lead to sub-optimal solutions. Consider a system as depicted in Fig. 3 with four devices with different processing times for work packages as indicated by the different lengths of the bars and 12 equal-sized work packages. In Fig. 3a the work packages are distributed at runtime to the devices just on request basis. Unfortunately, device 3 gets assigned the last work package 12, since the other devices are still busy.

However, a better distribution of work packages exists as shown in Fig. 3b. Although device 3 finished processing and is idle, it should not request another work package, but leave it to faster devices. Thus a simple on-request work package scheduler is not sufficient and more sophisticated techniques are required.

Minimal energy. An optimization directed at achieving minimal energy consumption tends to "serialize" the execution of an application. Minimal energy consumption means that only the most energy-efficient devices are used. As a consequence, in an heterogeneous environment only one type of devices with best energy-efficiency will be used. If only one instance of that device type exists in a cluster, the application will be executed on one device only.

Minimal energy with restrictions to performance. Since the minimal energy approach yields an extreme solution, there is a need for a relaxed approach. One possibility is to define a maximal execution time which shall be met with minimal energy preventing in this way the "serializing" behavior. As shown in Fig. 4a the programmer sets a time constraint for which the most energy-efficient distribution shall be found. All distributions below the dotted line meet the time constraint with the circular point being the most energy-efficient one.



(a) Simple request-based mapping

(b) More sophisticated mapping

Fig. 3: Different mappings of work packages onto devices

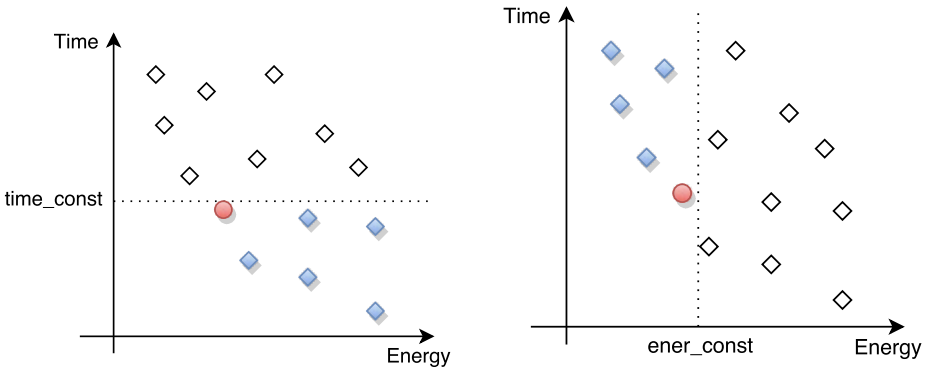
Best performance with restrictions to energy. Similar considerations as above lead to the definition of a maximal amount of energy which shall be met with best execution time possible. As shown in Fig. 4b the programmer sets an energy constraint for which the most time-efficient distribution shall be found. All distributions left to the dotted line meet the energy constraint with the circular point being the most time-efficient one.

Time-energy trade-off. Next we address the problem that no constraints have been specified by the programmer. Our goal is to propose a "good" solution by reasoning about time-energy trade-offs.

Fig. 5a shows the set of all possible distributions and Fig. 5b the corresponding Pareto front. It is reasonable to assume that a distribution of the Pareto front with minimal distance from the origin is a good compromise between execution time and energy consumption–the selected distribution is highlighted with a circular point.

5 Related work

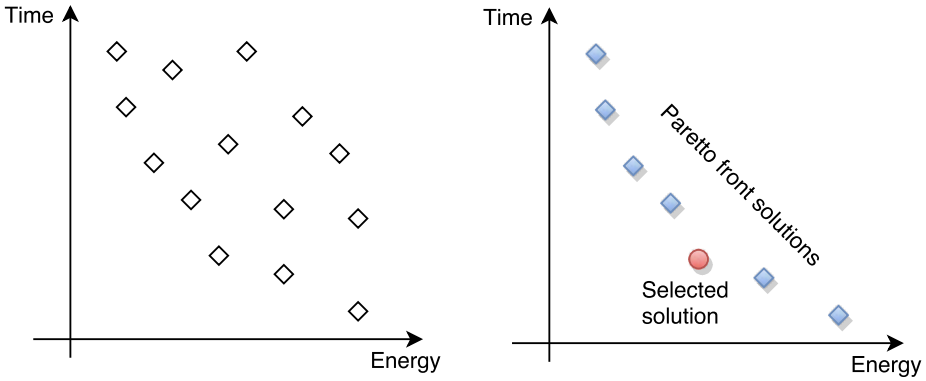
Many research efforts have been undertaken to support cluster systems, but most of them do not address optimizations taking both execution time and energy consumption into account in an heterogeneous, asymmetric hardware environment. One of the first cluster frameworks have been developed by Duato *et al.* [5] for CUDA and Barak *et al.* [3] for OpenCL. A more recent approach is SnuCL [8] which enables viewing of remote devices as part of a local context, while abstracting the communication between cluster nodes. libWater [6] is similar to



(a) Minimal energy with time constraint

(b) Best performance with energy constraint

Fig. 4: Solutions with constraints



(a) All possible distributions

(b) Pareto optimal distributions

Fig. 5: Set of possible solutions and Pareto front optimal solutions

the SnuCL approach, though it provides more efficient communication between cluster nodes, which is handled transparently by its runtime. Hybrid OpenCL [2] is based on FOXC runtime and enables communication between different OpenCL implementations in the context of distributed computing, while our framework targets HPC clusters. dOpenCL [7] allows using of different compute devices in any of the cluster nodes in a single application. It provides a central device manager which manages the assignment of the devices. clOpenCL [1] uses a wrapper library similar to dOpenCL targeting HPC clusters. DistCL [4] is similar to our approach, as it intends to abstract the multiple-devices in a cluster providing the programmer a single-device view for launching a kernel. However, DistCL runs its experiments across a symmetric cluster and only with one GPU per cluster. VOCL [13] provides a virtualization framework for GPU clusters using remote GPUs through proxy processes in cluster nodes. pVOCL [9] utilizes the VOCL framework and provides means for reducing energy consumption on a cluster. However, this approach is restricted to GPU clusters and it does not take into account actual energy consumption of applications but it is based on a table power model which needs to be provided by data center administrators. Whereas the cluster approaches presented so far have their origin in OpenCL, HeteroMPI [10] is an extension to MPI to support heterogeneous networks of computers. HeteroMPI provides the functionality to enable an application developer to deal with a heterogeneous environment and to realize a required behavior, but does not support the programmer with advanced features.

6 Conclusion

In this paper we discussed in detail the problem of distributing work onto devices of an heterogeneous, asymmetric cluster when both execution time and energy consumption should be taken into account. We presented different distribution strategies and discussed their implications in practice.

References

1. Alves, A., Rufino, J., Pina, A., Santos, L.: clOpenCL - Supporting Distributed Heterogeneous Computing in HPC Clusters. In: Euro-Par 2012: Parallel Processing Workshops, LNCS, vol. 7640, pp. 112–122. Springer Berlin Heidelberg (2013)
2. Aoki, R., Oikawa, S., Nakamura, T., Miki, S.: Hybrid OpenCL: Enhancing OpenCL for Distributed Processing. In: International Symposium on Parallel and Distributed Processing with Applications (ISPA). pp. 149–154 (May 2011)
3. Barak, A., Ben-Nun, T., Levy, E., Shiloh, A.: A package for OpenCL based heterogeneous computing on clusters with many GPU devices. In: 2010 IEEE Conference on Cluster Computing Workshops and Posters. pp. 1–7 (Sept 2010)
4. Diop, T., Gurfinkel, S., Anderson, J., Jerger, N.: DistCL: A Framework for the Distributed Execution of OpenCL Kernels. In: IEEE Symposium MASCOTS. pp. 556–566 (Aug 2013)
5. Duato, J., Pena, A., Silla, F., Mayo, R., Quintana-Orti, E.: rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. In: HPCS. pp. 224–231 (June 2010)

6. Grasso, I., Pellegrini, S., Cosenza, B., Fahringer, T.: LibWater: Heterogeneous Distributed Computing Made Easy. In: ICS. pp. 161–172. Eugene, Oregon (2013)
7. Kegel, P., Steuwer, M., Gorlatch, S.: dOpenCL: Towards a Uniform Programming Approach for Distributed Heterogeneous Multi-/Many-Core Systems. In: Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW). pp. 174–186 (May 2012)
8. Kim, J., Seo, S., Lee, J., Nah, J., Jo, G., Lee, J.: SnuCL: An OpenCL Framework for Heterogeneous CPU/GPU Clusters. In: Proceedings of the 26th ACM International Conference on Supercomputing (ICS). pp. 341–352. San Servolo Island, Venice, Italy (2012)
9. Lama, P., Li, Y., Aji, A.M., Balaji, P., Dinan, J., Xiao, S., Zhang, Y., Feng, W.c., Thakur, R., Zhou, X.: pVOCL: Power-Aware Dynamic Placement and Migration in Virtualized GPU Environments. In: Proceedings of ICDCS. pp. 145–154. Philadelphia, USA (2013)
10. Lastovetsky, A., Reddy, R.: HeteroMPI: Towards a message-passing library for heterogeneous networks of computers. Journal of Parallel and Distributed Computing 66(2), 197 – 220 (2006)
11. Liu, Q., Luk, W.: Heterogeneous systems for energy efficient scientific computing. In: Reconfigurable Computing: Architectures, Tools and Applications, LNCS, vol. 7199, pp. 64–75. Springer Berlin Heidelberg (2012)
12. Raca, V., Mehofer, E.: Device-Sensitive Framework for Handling Heterogeneous Asymmetric Clusters Efficiently. In: 26th IEEE International Symposium on Computer Architecture and High Performance Computing. Florianopolis, Brazil (Oct 2015)
13. Xiao, S., Balaji, P., Zhu, Q., Thakur, R., Coghlan, S., Lin, H., Wen, G., Hong, J., chun Feng, W.: VOCL: An optimized environment for transparent virtualization of graphics processing units. In: InPar. pp. 1–12 (May 2012)