2011 International Conference on Parallel Processing

Implications of Merging Phases on Scalability of Multi-core Architectures

Madhavan Manivannan* Ben Juurlink† Per Stenstrom*
Chalmers University of Technology*
Technische Universität Berlin†
{madhavan,per.stenstrom}@chalmers.se
{b.juurlink}@tu-berlin.de

***Abstract*—Amdahl's Law dictates that in parallel applications serial sections establish an upper limit on the scalability. Asymmetric chip multiprocessors with a large core in addition to several small cores have been advocated for recently as a promising design paradigm because the large core can accelerate the execution of serial sections and hence mitigate the scalability bottlenecks due to large serial sections.**

This paper studies the scalability of a set of data mining workloads that have negligible serial sections. The formulation of Amdahl's Law, that optimistically assumes constant serial sections, estimates these workloads to scale to hundreds of cores in a chip multiprocessor (CMP). However the overhead in carrying out merging (or reduction) operations makes scalability to peak at lesser number. We establish this by extending the Amdahl's speedup model to factor in the impact of reduction operations on the speedup of applications on symmetric as well as asymmetric CMP designs. Our analytical model estimates that asymmetric CMPs with one large and many tiny cores are only optimal for applications with a low reduction overhead. However, as the overhead starts to increase, the balance is shifted towards using fewer but more capable cores. This eventually limits the performance advantage of asymmetric over symmetric CMPs.

***Index Terms*—Amdahl's Law; Redcution operations; Chip Multiprocessor**

I. Introduction

Although technology roadmaps of CMPs (or multi-cores) predict a doubling of the number of cores with each new generation, it is not clear what types of cores should populate future chips. To aid in an early design exploration phase, architects use Amdahl's Law [1]. It states: If a fraction $1 - s$ of a sequential application can be parallelized, speedup in the limit will approach $\frac{1}{s}$, i.e. it is limited by the serial section in the parallel implementation.

In the context of CMPs, Amdahl's Law paints a dark future for how to leverage their performance. Even well-tuned parallel applications with a serial section that comprises only one percent will face a scalability limit at around one hundred cores. Hill and Marty [6] predict the scalability limit of asymmetric CMPs (ACMPs) [12], [11] where one core is optimized to accelerate serial sections and the rest of the chip hosts as many cores as possible to accelerate the parallel section. While they find that such ACMPs outperform CMPs, serial sections may still limit scalability. More recently, it was shown that such ACMPs do not yield optimal speedup for applications with large critical sections because of the inability of the small cores to execute the serializing critical sections efficiently [4]. These applications provide better scalability with fewer, but more capable, cores in place of maximizing the number of cores that fit on the chip.

Data mining is an emerging application domain [3] that can benefit from chip multiprocessors owing to its inherent parallel nature. In this paper, we focus on to what extent serial sections in these applications limit the scalability and how their characteristics influence the design of future CMPs. Our study is based on the clustering applications in the MineBench [13] suite. Since these applications have small serial sections (typically less than 0.1%), in accordance to Amdahl's Law we would expect them to scale to hundreds of cores. However, we surprisingly find that scalability is seriously hampered by reduction operations in the merging phase.

The merging phase typically assembles partial results from a parallel section and has an inherently serial component and can be found in other applications than the ones we study [5], [8]. We instrument the studied applications from MineBench and find that although the serial sections comprise only a fraction of a percentage to start with, the time spent on serial sections grows as we increase the number of cores and hence provides a lower theoretical estimate than Amdahl's Law. We extend the analytical model [6] to factor in the overhead of merging phases and use it to model the scalability of data mining applications on CMPs as well as ACMPs.

The main contributions of this study are the following: We extend and validate Amdahl's model for estimating speedup limits taking merging phases into account. Our extensions to the speedup model shows that, contrary to what Amdahl's Law predicts, speedup peaks at a much lesser core count. This has several interesting implications for CMP design. Firstly, we notice that due to the observed reduction overhead, ACMPs with a few powerful cores and many tiny cores may not be warranted; instead our study shows that reduction overhead pushes for a design with fewer and more capable cores. Secondly, we also show that reduction overhead limits the performance potential of ACMPs over CMPs.

Section II introduces Amdahl's Law and merging phases. Section III discusses extensions to incorporate the effect of merging phases. Section IV presents the evaluation methodology and our results are presented in Section V. Section VI puts our work in context to other work before we conclude.

0190-3918/11 $26.00 © 2011 IEEE
DOI 10.1109/ICPP.2011.74

II. Background

A. Amdahl's Law

The mathematical formulation of Amdahl's Law can be used to evaluate the speedup of a parallel application. Assuming that s represents the fraction of serial execution and f the fraction of parallel execution, where $s + f = 1$, the maximum achievable speedup with p processors is:

$$speedup_{limit} = \frac{1}{s + \frac{f}{p}} \quad (1)$$

which in the limit will approach $\frac{1}{s}$. Hill and Marty [6] evaluate the optimal core size to get the highest speedup out of a CMP: Assuming that a chip at a given time can host 256 *base-core equivalent* (BCE), they compare several possible design alternatives; 256 cores each consuming 1 BCE against 64 cores each consuming 4 BCEs, for instance. In general, in their model, CMPs can be built from $\frac{n}{r}$ cores where a total of n BCEs can be hosted on the chip and each core consumes r BCEs. Assuming that the performance of a core with r BCEs is $perf(r)$ greater than that of a single BCE, their extended model for computing speedup on CMPs is as follows:

$$speedup_{cmp} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f.r}{perf(r).n}} \quad (2)$$

They find that as the serial fraction increases, it will tend to favor designs with fewer and more capable cores. For asymmetric architectures they assume a large core that consumes r BCEs and $n - r$ small cores, each consuming one BCE. The expression for speedup assuming an ACMP is as follows:

$$speedup_{acmp} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}} \quad (3)$$

The mathematical formulation of Amdahl's law in Equations 1, 2 and 3 assumes that the serial section remains constant, independent of scaling. As will be shown in the following sections, this assumption is optimistic and tends to overestimate scalability.

B. Reduction Operations

We illustrate reduction operations (Algorithm 1) by examining the merging phase in the kmeans clustering application from MineBench. The reduction operations that are part of this phase grow linearly with the number of cores and the models presented in Section II-A cannot capture this effect when predicting speedup. Although the snippet from the application shown implements reduction serially (linear), this can be implemented in logarithmic steps also. We will consider this in our analysis later. We extend the speedup model in Equations 2 and 3 to factor in the effect of reductions in the next section.

III. Extension of Amdahl's Model

We use the speedup expression for CMP architectures according to Equation 2 as baseline and make extensions to factor in the effect of reduction overhead operations. The serial fraction s in our model assumes that serial fraction is non-constant and is dependent on scaling as opposed to the assumption made by the formulation of Amdahl's Law stated in Section II-A. s comprises of the constant serial fraction (f_{con}), and the reduction fraction (f_{red}) where f_{con} represents the fraction of serial section time without considering reduction operations. The reduction fraction is further split into a constant reduction (f_{cred}) fraction and a reduction overhead (f_{ored}) fraction that determines the fraction of reduction computations that grows as we scale. The overhead fraction grows at a certain rate determined by the function $grow()$ in the expression. We study the impact of having $linear$ and $logarithmic$ growth functions. The split up of the serial fraction into its constituent parts is illustrated in Figure 1.

Algorithm 1 kmeans merging phase

```
for i = 1 → nclusters do
  for j = 1 → nthreads do
    new_centers ← += partial_centers
  end for
end for
```

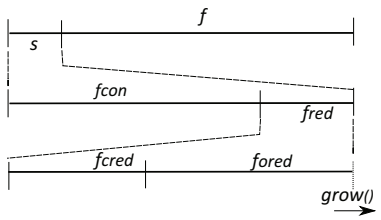


Fig. 1. Serial section splitup

The modified expression for computing the speedup for CMPs ($speedup_{sym}$) according to Equation 2 substitutes this formulation in place of $s = (1 - f)$ and is as follows:

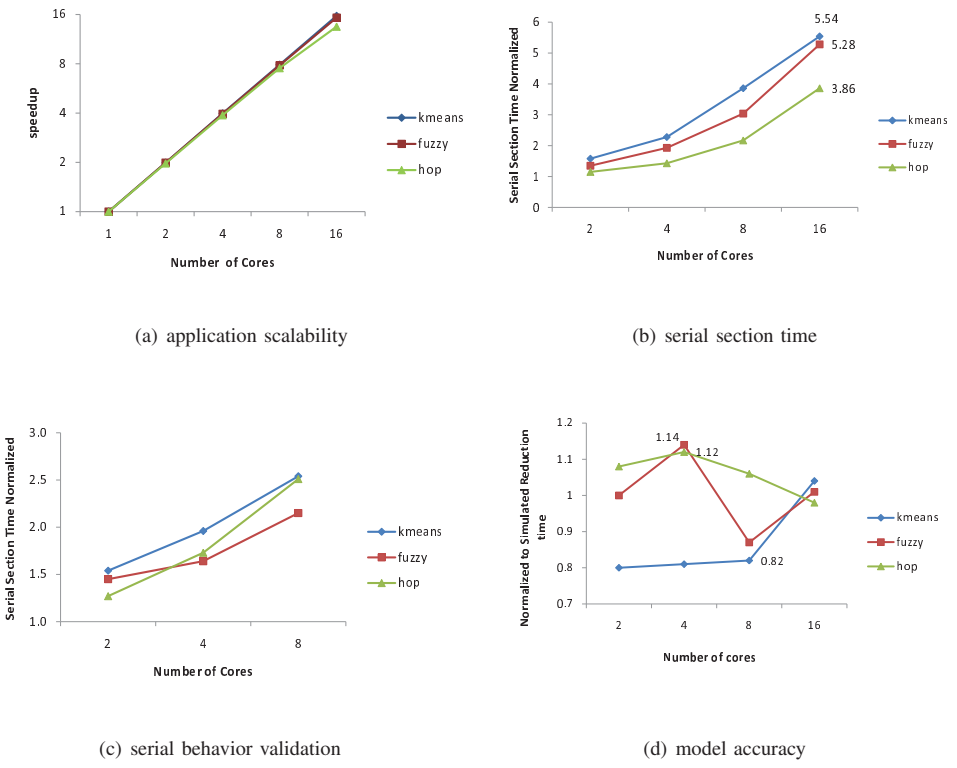
$$speedup_{cmp} = \frac{1}{\frac{f_{con}+f_{cred}+f_{ored}.grow(n,r)}{perf(r)} + \frac{f.r}{perf(r).n}} \quad (4)$$

Speedup for ACMPs ($speedup_{acmp}$) is obtained by introducing a new parameter in the expression that models the effect of introducing a large (r_l BCEs) core in the design. This expression reflects the impact of executing the serial section on the large core and the parallel section on $\frac{n-rl}{r}$ cores with performance $perf(r)$ (and one large core) instead of homogeneous cores.

$$\frac{1}{\frac{f_{con}+f_{cred}+f_{ored}.grow(n,r,r_l)}{perf(r_l)} + \frac{f}{perf(r).\frac{n-r_l}{r}+perf(r_l)}} \quad (5)$$

IV. Evaluation Methodology

We use the SESC [14] simulator to simulate the execution of the applications on a chip multiprocessor platform in order to extract the application parameters required for the analytical model. The architecture we simulate consists of a number of cores using a private L1 cache and a shared L2 cache. We limit the number of cores in the simulations to 16 as more cores leads to unreasonably long simulation times. The



(a) application scalability

(b) serial section time

(c) serial behavior validation

(d) model accuracy

Fig. 2. Application characterization

baseline architecture parameters are enlisted in Table I. We use all the multi-threaded clustering benchmarks (kmeans, fuzzymeans and hop) from Minebench [13] with default data set, because they have the following characteristics a) They are all scalable (high parallel fraction) and hence fit the simple Amdahl speedup model, expect for one specific kernel in hop b) These applications have low/no synchronization overheads. Inorder to study theimpact of scaling beyond 16 cores, we use the analytical model derived in Section III to predict scalability. We also validate our observation by extracting the parameters on real hardware using a two-socket machine with Xeon E5520 CPU (four core CPU) comprising a total of 8 cores with 24GB RAM. The application source code was slightly modified to use a different threading library due to compatibility issues with the simulator. These modifications do not influence the serial section behavior.

TABLE I
BASELINE CONFIGURATION

Fetch, Issue, Commit	4
Instn. Window, LSQ, ROB	32, 16, 64
L1 I/D Cache, L2 Cache, Coherence	16K/64K 2/4 way private, 4M 16 way shared, MESI
Branch Pred., BTBSize	2level GAp 2048 entr., 512

V. RESULTS

A. Impact of Reduction Operations

We first examine the scalability offered by the clustering applications using simulation data. Figure 2(a) plots the scalability offered by the applications as we scale them to 16 cores. From the graph we can observe that kmeans and fuzzy scale linearly up to 16 cores as they exhibit a speedup close to 16 while hop shows moderate speedup of around 13.5. Hop does not scale well mainly because the parallel tree construction kernel does not scale up to 16 cores.

The close-to-linear speedup suggests that the serial sections in the applications are small. The time spent on the serial section using the simulation infrastructure presented in Section IV is first obtained by counting the total cycles spent in the application and subtracting from it the time spent on parallel sections and initialization. We then divide this by the total execution time on a single core and obtain the serial fraction. In doing so, we observe that the serial sections indeed occupy a small fraction of the application and consume between 0.002% and 0.1% of the entire application execution time (see Table II).

Critical sections are potentially another source of serialization [4] and it is therefore important to understand their impact in the context of serial sections. We measure the fraction of time spent in critical sections when each application is run on a single processor (see Table II). Since critical

TABLE II
APPLICATION PARAMETERS

Application	serial (%)	critical section (%)	f_{ored} (%)	f_{red} (%)	f_{con} (%)	f
kmeans	0.015	0.004	72	43	57	0.99985
fuzzy	0.002	0	82	35	65	0.99998
hop	0.100	0.0003	155	12	88	0.99900

sections account for a very small fraction of the sequential execution time, it is fair to assume that the serialization they cause will be negligible. Owing to the difficulty associated with measuring the fraction of time spent on critical sections accurately without introducing any timing perturbations we exclude them in our analysis of serial sections.

Since the formulation of Amdahl's Law optimistically assumes constant serial sections, it predicts that these applications would scale to large number of cores. However, we observe that the actual amount of time spent on serial sections grows as we increase the number of cores. Figure 2(b) shows the time spent in executing serial sections s on multiple cores normalized to the time spent executing the serial section on a single core. For all these applications, the serial section time (constituted by constant serial section and reduction section) grows significantly with the number of cores as opposed to remaining constant owing to reduction overhead.

We now consider what implications this has on scalability using the analytical model presented in the paper. The values for the different parameters that are required for the analytical model presented in Section III are obtained through simulation by timing the individual sections of the application. f_{con} is obtained by measuring the time spent in the serial section without taking into account the reduction operations, f_{cred} is obtained by measuring the time spent on reduction when the application only uses a single core and f_{ored} is obtained by measuring the relative increase in reduction operation time over fcred when using multiple cores. The parameters for kmeans, fuzzy and hop are presented in Table II and are obtained using the simulation infrastructure discussed in the previous section. f_{con}, f_{cred} and f_{ored} are expressed as fractions of serial time in the table. For instance, in kmeans the reduction value presented in the table indicates that 43% of the time spent on executing serial section is actually spent on reduction operations. The applications presented in Table II follows a linear growth function. In case of hop, we notice that the overhead grows superlinearly with core count and believe that this is due to large number of memory accesses in the merging phase that adds to the assumed linear growth of the merging phase.

B. Model Validation

We first validate the observation about growing serial sections by running these applications on real hardware. We use the hardware infrastructure described in Section IV using large data sets and time the serial section. Figure2(c) plots the serial section time obtained as we scale the application normalized to serial section time on a single core. From the graph we can observe that all applications indeed exhibit a serial section growth characteristic similar with what is obtained from simulations, when run on real hardware.

We next validate the model presented in Section III to see if it matches with the simulation results that we obtain. In Figure 2(d), we normalize the serial section time predicted by the model with the time obtained through simulation to establish the accuracy by which our extended model can predict the growth in the serial section.

From the graph we can observe that in the case of kmeans our model mostly underestimates the impact due to the growth of the serial section and in the case of hop it mostly overestimates the impact and for fuzzy the estimation varies. The maximum overestimation margin is observed for fuzzy with 14% and the maximum underestimation margin is observed for kmeans with 18%. This demonstrates that the simple extensions proposed in Section III closely track the growing serial section behavior.

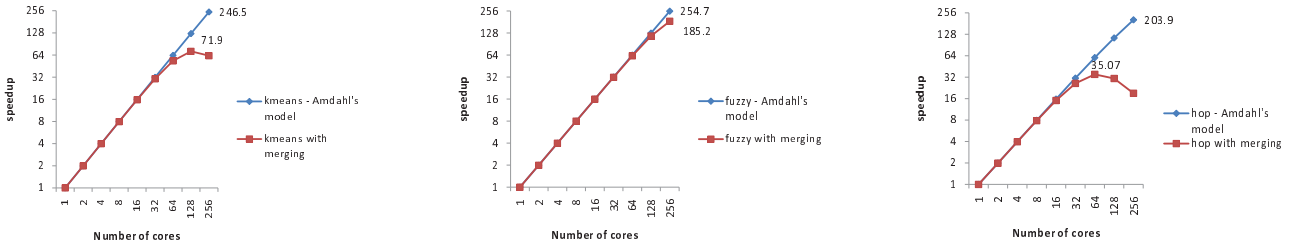
C. Impact on Scalability

We now compare the speedup predicted using the proposed model taking the impact of reduction operations into account against a model which does not and compare the predicted speedups as we scale to 256 cores. Both models assume that the parallel sections scale linearly with the number of cores. Figure 3 plots the scalability predictions for kmeans, fuzzy and hop using the parameters presented in Table II, with and without considering reduction overhead operations. For this analysis we assume the core presented in Table I as the baseline with a performance of unity and that the architecture consists of 256 such cores.

The graphs indicate that if the serial section is optimistically assumed to remain constant independent of scaling it would result in overestimation of scalability for such applications. Under the assumption that serial sections are constant, as the curve corresponding to Amdahl's model assume, speedup linearly scales to at least 256 cores. However, by factoring in growth of overhead of serial sections, speedup tapers off at much lesser core count. This observation is important as it shows that applications that have very high parallel fraction values can have serial sections which if properly factored in can prevent applications from scaling. This also goes to show that naively using Amdahl's Law can lead to speedup overestimation.

D. Implications for CMP Design

We categorize applications along three dimensions: Size of parallel section, how big fraction of serial section that is constant, and the impact of reduction overhead and consider their impact on speedup. For each dimension, we consider two cases. For the first dimension, we distinguish between embarrassingly parallel ($f = 0.999$) or non-embarrassingly parallel ($f = 0.99$) applications. As for the second dimension, we distinguish between a high constant fraction ($fcon(\%) = 90$) and a moderate constant fraction ($fcon(\%) = 60$). Finally,



(a) kmeans (b) fuzzy (c) hop

Fig. 3. Scalability prediction using different models

TABLE III
APPLICATION CLASSES AND PARAMETERS

parallelism	constant	reduction	f	f_{con} (%)	f_{ored} (%)
Emb.	high	low	0.999	90	10
Non-emb.	high	low	0.99	90	10
Emb.	moderate	low	0.999	60	10
Non-emb.	moderate	low	0.99	60	10
Emb.	high	high	0.999	90	80
Non-emb.	high	high	0.99	90	80
Emb.	moderate	high	0.999	60	80
Non-emb.	moderate	high	0.99	60	80

considering the third dimension we distinguish between low ($fored(\%) = 10$) and high reduction overhead ($fored(\%) = 80$). Since we emphasize on scalability bottlenecks that can influence application performance we only consider cases with high parallel fraction to start with. As for the other parameters we believe that we have used conservative numbers covering interesting parts of the design space rather than in entirety. The parameters are summarized in Table III.

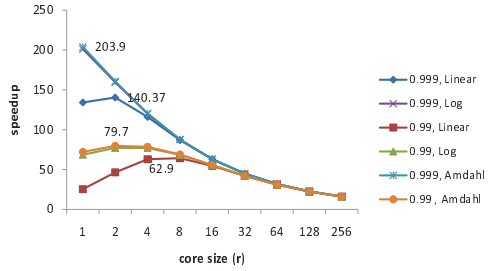
For the rest of the analysis we assume that the performance of a core is proportional to the square root of the area [2], i.e., a core made up of four BCEs performs twice as high as a single BCE. We further assume a hardware budget of 256 BCEs and a linear reduction overhead growth function unless specified otherwise.

*1) **Symmetric Chip Multiprocessors**:* We first analyze the impact of reduction operations on CMPs which comprise of homogeneous cores. Through this analysis, we try to determine a suitable architecture (many tiny cores or few large cores) that would yield maximum speedup for applications with reduction overhead operations. Figure 4 plots the speedup computed using Equation 4 in Section III and the parameters listed in Table III. The x-axis in the graph represents the amount of area allocated to a single core in a CMP: a value of 1 implies a design with 256 cores of 1 BCE each and a value of 4 implies 64 cores of 4 BCEs each. The y-axis in the graph plots the speedup of each configuration with respect to a single BCE.

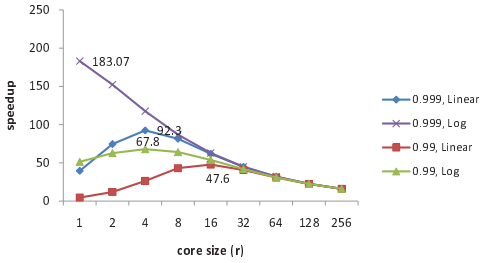
First we discuss results computed assuming a linear growth function (marked $Linear$). From the graphs we can observe that the highest speedups for the curves marked $Linear$ never occur if we scale to large number of cores, i.e., a design with 256 cores ($r = 1$ in the x-axis) never yields the highest speedup as shown in Figure 4(a) to 4(d). This is because the reduction overhead fraction grows as we scale up and this dwarfs the benefit of accelerating the parallel section. There is loss in scalability as the overhead fraction grows and fewer but larger cores are required to achieve the maximum speedup. This can be seen by taking note of where speedup peaks when moving from graphs 4(a) and 4(c), which correspond to low reduction overhead, to 4(b) and 4(d), which correspond to high reduction overhead. This trend can be observed for embarrassingly parallel as well as non-embarrassingly parallel applications, irrespective of the amount of constant fraction. Moreover there is also a drop in maximum speedup. For instance (0.999, $Linear$) in graph 4(c) attains a maximum speedup of 104.5 for $r = 4$ whereas in graph 4(d) maximum speedup of 67.1 is attained for $r = 8$.

For reduction overhead operations with logarithmic growth (marked Log in the graphs) we observe that the scalability trend discussed previously still applies for non-embarrassingly parallel applications. For embarrassingly parallel applications, however, small cores manage to yield the highest speedup. Applications with a high reduction overhead, however, still experience drop in scalability. These observations collectively suggest that CMPs with large cores are generally required to execute applications with high reduction overhead fractions efficiently. For such applications we will have to accommodate fewer but larger cores and thereby trade off performance for applications that have potential for effectively using large number of cores.

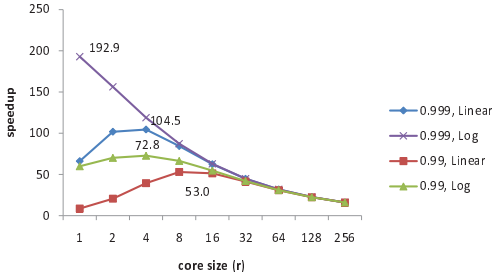
*2) **Asymmetric Chip Multiprocessors**:* We now focus on the impact of reduction overhead on ACMPs which consist of one large core that executes the serial sections and several homogeneous cores for executing the parallel sections. The interesting question is whether such ACMPs have a performance advantage over CMPs in presence of reduction operations. Figure 5 plots the speedup computed using Equation 5 in Section III for ACMPs for varying r_l and r. The x-axis in the graph represents the area allocated to the large core in an ACMP and the y-axis in the graph plots the speedup of each configuration with respect to a single BCE. The different



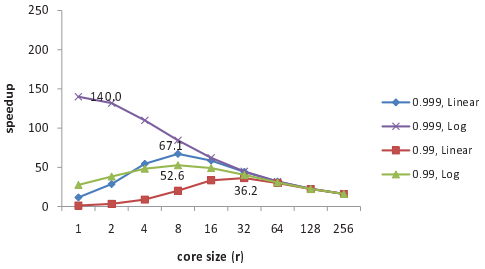
(a) High constant, low reduction overhead



(b) High constant, high reduction overhead



(c) Moderate constant, low reduction overhead



(d) Moderate constant, high reduction overhead

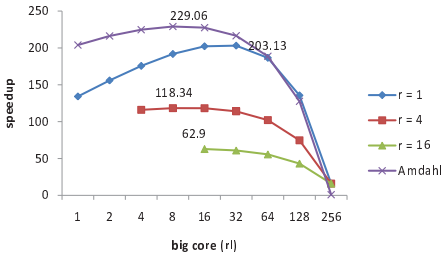
Fig. 4. Scalability on symmetric CMPs

lines ($r = 1, 4, 16$) in each graph represent the amount of resources allocated to each of the homogeneous cores that are responsible for the parallel sections. We assume that the reduction operations only happen on the large core in the design (linear complexity).

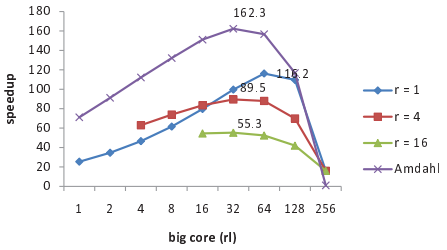
For applications with low reduction overhead, such as in graph 5(a), 5(b), 5(e) and 5(f), maximizing the number of cores for the parallel section in addition to the size of the large core always yields maximum speedup. We can observe this for embarrassingly parallel as well as non-embarrassingly parallel applications irrespective of the amount of constant fraction. In all these graphs the maximum speedup is achieved for the plot $r = 1$ BCE and this indicates that having many small cores for parallel sections in addition to one large core for the serial section would yield maximum speedup for such applications. Embarrassingly parallel applications with high reduction overhead show a similar trend irrespective of their group as seen in graphs 5(c) and 5(g).

For non-embarrassingly parallel applications that exhibit a high reduction overhead and with a high constant fraction, as in Figure 5(d), using more capable cores for executing parallel sections yields maximum speedup ($r = 4$ BCEs yields higher speedup than $r = 1$ BCE). ACMPs still manage to provide good improvement in speedup over CMPs for such applications. For applications in this category CMPs (Figure 5(b)) yield a maximum speedup of 47.6 whereas ACMPs yield a speedup of 64.2.

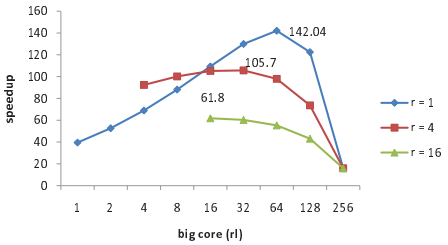
For non-embarrassingly applications that have a high reduction overhead and a moderate constant fraction, ACMPs that use many small cores apart from the large core for handling parallel sections, as in Figure 5(h) for the case $r = 1$, perform worse (speedup = 22.6) than symmetric designs as in Figure 4(d) (speedup = 36.2 for $Linear$ under $f = 0.99$), contrary to the predictions using Amdahl's Law (speedup = 162.3 vs. 79.7 for the asymmetric and symmetric case, respectively) . This is because the reduction overhead tends to dwarf the benefit of having many small cores to scale the parallel section. Designs with fewer more capable cores also do not yield significant performance improvement over CMPs. For instance, in considering the same application case, ACMPs yield a maximum speedup (Figure 5(h)) of 43.3 ($r = 4$) and symmetric architectures yield a maximum speedup (Figure 5(d)) of 36.2 ($r = 32$). In general, a large reduction overhead limits the performance benefit of asymmetric designs over



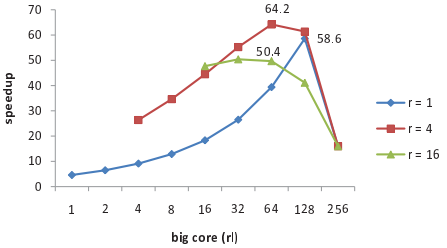
(a) Emb., high constant, low reduction overhead



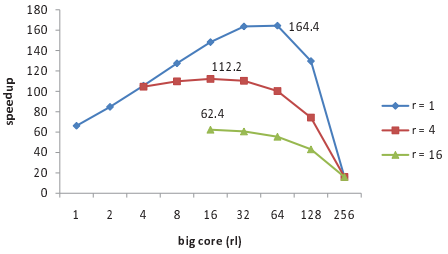
(b) Non-emb., high constant, low reduction overhead



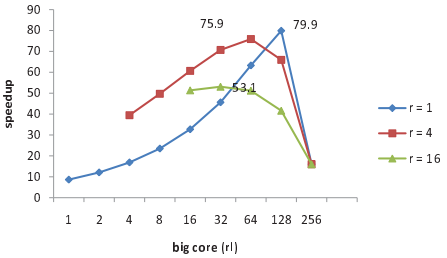
(c) Emb., high constant, high reduction overhead



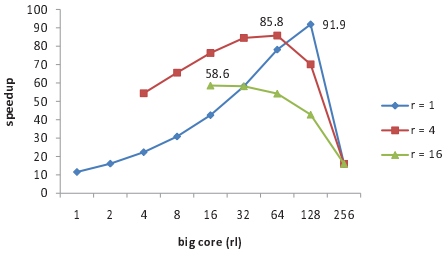
(d) Non-emb., high constant, high reduction overhead



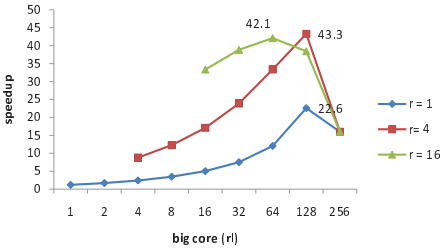
(e) Emb., moderate constant, low reduction overhead



(f) Non-emb., moderate const., low reduction overhead



(g) Emb., moderate constant, high reduction overhead



(h) Non-emb., moderate const., high reduction overhead

Fig. 5. Scalability on asymmetric CMPs

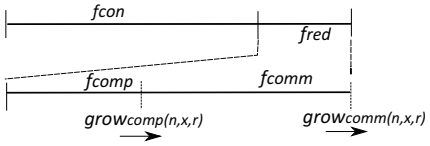


Fig. 6. Reduction fraction splitup

symmetric designs.

These results suggest that applications with low reduction overhead require a large core to efficiently execute serial sections. In such a scenario, ACMPs can provide performance benefits over CMPs. As the overhead increases there is a shift towards fewer and more capable cores. Additionally, the results obtained by our extended analytical model suggest that the benefit of ACMPs over CMPs is indeed quite limited for applications with large overhead.

E. Impact of Commmunication

In the previous section it is assumed that work performed in the reduction phase always grows linearly with the number of cores. The model however does not consider the impact of communication although it is a critical factor in the merging phase. In a typical merging phase where results computed across different threads are accumulated, the computations have little overhead when compared to the communication operations that need to be performed. In this section we extend the model to cover this. In addition to the linear and logarithmic reduction costs discussed previously we now consider the impact of having a parallel implementation of merging phase. In a parallel reduction scenario that involves x reduction elements from n cores, each thread is assigned a part of the reduction ($\frac{x}{n}$) elements. The computation costs associated with parallel implementation does not scale ($\frac{x}{n}.n = x$) with the number of cores unlike the other approaches. The communication cost associated with transferring the partial cluster information computed by the other threads to the master core however grows by $(n-1).x$ and this is because each core needs to send (receive) a subset of privatized reduction elements to (from) all the other cores. Assuming a common case it grows by $2.(n-1).x$ because the reduction results also need to be broadcasted back to other cores.

We use the speedup expressions presented in Section III as baseline and make extensions to factor in the effect of communication on reduction operations. Instead of simply assuming that the reduction fraction is constituted by a constant reduction (f_{cred}) fraction and an overhead reduction (f_{ored}) fraction as discussed before we consider that it is constituted by a computation fraction (f_{comp}) and a communication fraction (f_{comm}). We assume an ideal case for communication where merging phase is equally represented by the communication and computation fraction. This is based on the premise that for reductions to happen the number of communication and computation operations remains the same assuming a single thread (f_{comp} == f_{comm} and $f_{comp} + f_{comm} = f_{red}$). We also do not consider overheads due to memory accesses because it favors f_{comm}. The term f_{comm} specifically represents the fraction spent on communicating x reduction elements and f_{comp} the fraction spent on computing reduction across x elements. The reduction fraction split-up is illustrated in Figure 6. The overheads associated with both fractions are represented as $grow_{comp}()$ and $grow_{comm}()$. The serial part in the speedup expression considering CMP is

$$\frac{f_{con} + f_{comp}.(1 + grow_{comp}(n,x,r)}{perf(r)} + \\ + f_{comm}.(1 + grow_{comm}(n,x,r)) \quad (6)$$

and considering ACMP is

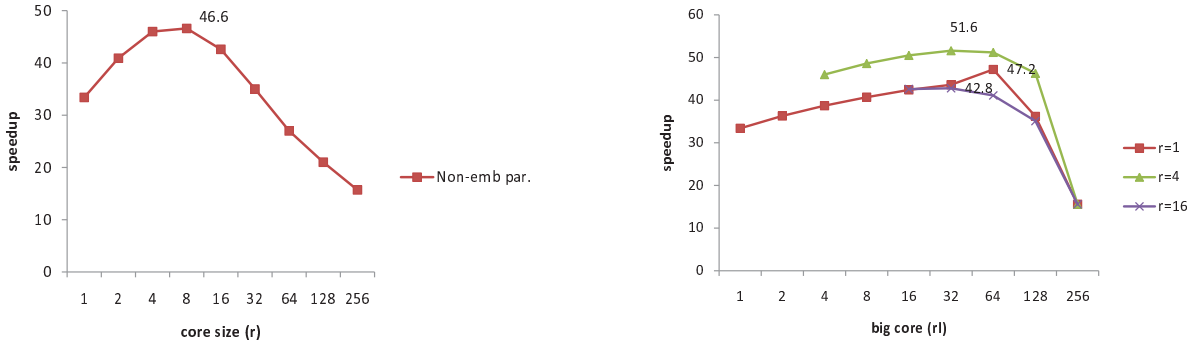
$$\frac{f_{con} + f_{comp}.(1 + grow_{comp}(n,x,r,r_l)}{perf(r_l)} + \\ + f_{comm}.(1 + grow_{comm}(n,x,r,r_l)) \quad (7)$$

It is evident that a linear reduction technique would have a linear computational cost function, a logarithmic technique would have a logarithmic computational cost function and a parallel technique would have no additional computational costs. However for communication operations, this depends on the number of operations that need to be performed and the number of operations that a given interconnection network can achieve in any given instance. For this we assume 2D mesh topology as it is the most commonly used topology in many core CMP studies. In case of a 2D mesh topology with n_c cores ($\frac{n}{r}$ for symmetric case and $\frac{n-r_l}{r+l}$ for asymmetric case) there exist $2.\sqrt{n_c}.(\sqrt{n_c}-1)$ links. Assuming bi-directional links we can establish $4.\sqrt{n_c}.(\sqrt{n_c}-1)$ operations at any given instance. A communication operation implemented over 2D incurs additional overhead due to latencies for each hop on the network. Since the packets take $(\sqrt{n_c}-1)$ hops on an average over the network before it reaches its destination, the total overhead due to communication is the product of the total number of communication operations and the average number of hops each operation travels over the network (assuming each hop takes unit time) $2.(n_c-1).x.(\sqrt{n_c}-1)$. The expression for $grow_{comm}()$ for a 2D mesh topology is hence as follows.

$$\frac{2.(n_c-1).x.(\sqrt{n_c}-1)}{4.\sqrt{n_c}.(\sqrt{n_c}-1)} \approx \frac{\sqrt{n_c}}{2} \quad (8)$$

In Equations 6 and 7 the expression that incorporates the influence of communicating reduction elements (Equation 8) is considered in addition to computation cost to estimate speedup for CMPs (Figure 7(a)) and ACMPs (Figure 7(b)). These graphs have axis similar to Figure 4 and Figure 5 and are plotted for non-embarassingly parallel application with a moderate constant fraction as shown in Table III.

From CMPs we notice that ($r = 8$ in the x-axis) yields the highest speedup which implies preference towards designs with fewer larger cores. Compared to the speedup suggested by Amdahl's model as shown in Figure 4(a) the estimated speedup is less (79.7 against 46.6). In case of asymmetric



(a) Scalability – symmetric CMPs

(b) Scalability – asymmetric CMPs

Fig. 7. Scalability using symmetric and asymmetric CMPs

CMPs we can observe the following. Firstly the maximum speedup estimate is 51.6 which is considerably lower than the speedup estimate provided by Amdahl's model (162.3 shown in Figure 5(b)). Secondly a design with fewer larger cores provides a slightly better estimate than having one large core and several small cores, although the margin is not significant ($r = 4$ provides greater estimate than $r = 1$). Finally we can observe from the graphs that the speedup improvement of ACMP over CMP is diminished. The grow function for this model remains to be validated and we will consider that for our future work. We however believe that since this model is based on ideal assumptions, it still provides an optimistic estimate of the trends when incorporating communication overhead.

F. Variational Analysis on Datasets

The application parameters presented in Table II have been derived from data sets whose size yield execution times that are reasonable for architecture-level simulations. This section aims at validating these parameters for larger data sets. For this, we consider multiple scaled data sets used by the clustering algorithms kmeans and fuzzy For hop, on the other hand, we just simulate the $default$ and the $medium$ data sets because the overhead associated with simulation is significant. The attributes that we are specifically focusing on for kmeans and fuzzy data sets are the Number of Points (N), Number of Dimensions (D), and Number of Cluster Centers (C) and the values are discussed in Table IV. The $default$ data sets to derive application parameters in the previous section are labeled as $kmeans-base$, $fuzzy-base$, and $hop-default$. The impact of data set size on the measured parallel fraction (f), constant serial fraction (f_{con}), and the reduction fraction (f_{red}) are listed.

From the table we can observe that for kmeans and fuzzy even after scaling the number of dimensions and the number of centers the parallel fraction either remains more or less same. Scaling the number of points, on the other hand, slightly increases the parallel fraction and this is because the number of operations performed in the merging phase is independent of the number of points and is only dependent on the number of dimensions and the number of clusters. Scaling the number of points in the data set only causes the work in the parallel section to increase with no change in the number of operations performed in the merging phase. For hop as we scale from the $default$ data set to the $medium$ data set the parallel fraction decreases. These results imply that even with larger datasets, we would expect to observe similar fraction values in most cases.

TABLE IV
DATASET SENSITIVITY

Data Label	Attributes	f	f_{red} (%)	f_{con} (%)
kmeans-base	N:17695 D:9 C:8	0.99985	43	57
kmeans-dim	N:17695 D:18 C:8	0.99984	41	59
kmeans-point	N:35390 D:18 C:8	0.99992	49	51
kmeans-center	N:17695 D:18 C:32	0.99984	41	59
fuzzy-base	N:17695 D:9 C:8	0.99998	65	35
fuzzy-dim	N:17695 D:18 C:8	0.99997	61	39
fuzzy-point	N:35390 D:18 C:8	0.99999	59	41
fuzzy-dim	N:17695 D:18 C:32	0.99998	61	39
hop-default	default 64p 61440	0.9990	12	88
hop-med	med 128p 491520	0.9980	15	85

VI. Related Work

There has been a considerable body of work focusing on asymmetric architectures and their ability to provide higher performance than symmetric architectures [6], [15], [12], [11]. Stijn and Eeckhout address the limitations of asymmetric CMPs by focusing on the impact of critical sections by incorporating the overheads associated with synchronization in Amdahl's Law [4]. In their analysis they assume that the effect of merging the data produced by multiple threads as a part of serial sections and they consider it to be a constant. We clearly demonstrate that this is not a constant but rather grows and limits scalability severely. Our model is mainly applicable to parallel applications that involve little or no synchronization

overhead. Such applications may also involve a merging phase where data worked upon by the different threads are assembled to make a final result. The work here is orthogonal to what they propose and these can combined along to improve accuracy of scalability prediction and the design insights.

Work has also targeted optimizing reduction operation execution for improving performance. Although there are several classes of reduction operations, we specifically focus our attention on partial write reductions [9]. Gagan et al. [9] establish that such classes of reduction operations are common across many categories of data mining applications in addition to the ones that we use in our analysis. They also provide detailed analytical models to determine the performance of each technique on a shared memory machine [8]. They however do not analyze the impact of reduction operations on CMP design choices. Garzaran et al. [5] propose architectural support for effectively carrying out such reduction operations. We however are not aware of any commercial microprocessor with specialized support for carrying out reductions and through this work want to highlight its importance. Holzle [7] briefly mentions embarrassingly parallel applications and the influence of computing stop criteria based on global information. He uses this to argue for symmetric architectures with few large cores instead of many small cores but does not provide any quantitative analysis and also does not discuss asymmetric architectures. Loh [10] models the cost of uncore resources in addition to core resources but does not consider the serializing nature of merging phases and its impact on scalability.

VII. Conclusions

Amdahl's Law estimates that applications with significant parallel fractions potentially scale to large number of cores. Surprisingly, we find that because of the reduction operations the serial sections in these applications do not remain constant anymore. We then extend Amdahl's Law to incorporate the effect of reduction operations and validate how accurately the model predicts the overhead of reduction operations against simulation on a CMP. Our most important findings using the proposed model are a) Amdahl's Law can overestimate the scalability offered by symmetric and asymmetric architectures for applications with merging phase and b) There is a shift towards using the chip area for fewer and hence more capable cores rather than simply increasing the number of cores for symmetric as well as asymmetric architectures and c) The performance potential of asymmetric over symmetric CMPs is limited for such applications.

Acknowledgment

This research has been supported by the Swedish Research Council (VR). The simulations were performed on C3SE computing resources. The authors thank Anurag Negi for his insightful comments on this work.

References

[1] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67, pages 483–485, 1967.

[2] Shekhar Borkar. Thousand core chips: a technology perspective. pages 746–749, 2007.

[3] Pradeep Dubey. A platform 2015 model: Recognition, mining and synthesis moves computers to the era of tera. *Intel Technology Journal*, February 2005.

[4] Stijn Eyerman and Lieven Eeckhout. Modeling critical sections in amdahl's law and its implications for multicore design. In *International Symposium on Computer Architecture*, pages 362–370, 2010.

[5] Mara Jess Garzarn, Milos Prvulovic, Ye Zhangy, Josep Torrellas, Alin Jula, Hao Yu, and Lawrence Rauchwerger. Architectural support for parallel reductions in scalable shared-memory multiprocessors. In *International Conference on Parallel Architectures and Compilation Techniques*, 2001.

[6] Mark D. Hill and Michael R. Marty. Amdahls law in the multicore era. *IEEE Computer*, July 2008.

[7] Urs Holzle. Brawny cores still beat wimpy cores, most of the time. *IEEE Micro*, July 2010.

[8] Ruoming Jin and Gagan Agrawal. A methodology for detailed performance modeling of reduction computations on smp machines. *Performance Evaluation*, 60:73–105, 2005.

[9] Ruoming Jin, Ge Yang, and Gagan Agrawal. Shared memory parallelization of data mining algorithms: Techniques, programming interface, and performance. *IEEE Transactions on Knowledge and Data Engineering*, 17:71–89, 2005.

[10] Gabriel H. Loh. The cost of uncore in throughput-oriented many-core processors. In *In Proc. of Workshop on Architectures and Languages for Troughput Applications (ALTA)*, 2008.

[11] Daniel Menascé and Virgílio Almeida. Cost-performance analysis of heterogeneity in supercomputer architectures. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, Supercomputing '90, pages 169–177, 1990.

[12] Tomer Y. Morad, Uri C. Weiser, Avinoam Kolodny, Mateo Valero, and Eduard Ayguade. Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. *IEEE Comput. Archit. Lett.*, 5, January 2006.

[13] Ramanathan Narayanan, Berkin Ozisikyilmaz, Joseph Zambreno, Gokhan Memik, and Alok Choudhary. Minebench: A benchmark suite for data mining workloads. In *2006 IEEE International Symposium on Workload Characterization*, pages 182–188, 2006.

[14] Jose Renau, Basilio Fraguela, James Tuck, Wei Liu, Milos Prvulovic, Luis Ceze, Smruti Sarangi, Paul Sack, Karin Strauss, and Pablo Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[15] M. Aater Suleman, Onur Mutlu, Moinuddin K. Qureshi, and Yale N. Patt. Accelerating critical section execution with asymmetric multi-core architectures. In *Architectural Support for Programming Languages and Operating Systems*, pages 253–264, 2009.