Abstract

In this paper we investigate the speedup potential of asynchronous iterative algorithms over their synchronous counterparts for the special case of linear iterations.

The space of linear iterations of size two is explored by simulation and analytical methods. We find cases and conditions for high asynchronous speedups. However, averaging asynchronous speedups over the whole set of iteration matrices reveals that the domain of matrices with high speedups is not sizable and that asynchronous iterations perform no better than synchronous ones over the entire convergence space.

1. Introduction

Finding a solution \( x^* \) to the generic fixed point problem \( x = f(x), \ x \in \mathbb{R}^n \) is a common way to solve many problems, such as systems of equations or optimizations. Iterative methods compute successive approximations of the solution by executing the basic iterative step

\[
x_{k+1} = f(x_k)
\]

with \( x_0 \) an initial estimation of the solution \( x^* \). This computation is parallelized by partitioning the set of indices for the components of \( x \) into disjoint subsets and performing the independent computations on different processors simultaneously. Synchronous methods reproduce the exact same sequence of iterations as a sequential implementation by synchronizing the processors at each step. Asynchronous methods [1] remove synchronization overhead and allow a new cycle to start with obsolete information. The trade-off is often an increase in the number of iterations.

Several papers have attempted to address these performance aspects. Qualitative results for generic classes of mappings, such as monotone or contractions, exist [3,4]. Some tell us when asynchronous iterations converge faster, but not how much faster, while some others give crude estimates of the asynchronous convergence rate.

In this paper we use simulations and analytical models to investigate cases when asynchronous execution leads to a sizable decrease in the number of iterations required for convergence. We have performed an exhaustive analysis of linear iterations with two variables. The entire convergence domain of iteration matrices is sampled to yield over a quarter million cases. We compare the number of iterations needed in the synchronous and asynchronous cases, ignoring the communication and synchronization overheads.

Our simulations reveal a sharp decrease in the number of iterations in the asynchronous case for very small domains of matrices having a dominant eigenvalue close to -1. But, when averaging performance over the whole set of matrices, there is no advantage for asynchronous iterations. These observations are confirmed by our analytical model of asynchronous linear iterations with stochastic delays.

The rest of the paper is structured as follows. Section 2 contains background material. Sections 3 and 4 are devoted to the simulation and analytical models. Finally, Section 5 contains our conclusions.

2. Background

2.1. Deterministic Convergence Conditions

We briefly review the (a)synchronous convergence conditions for the linear iteration \( x = Ax + b \) [4,6,8].

**Synchronous Iterations:** they are described by

\[
x_{k+1} = Ax_k + b,
\]

where the integer \( k \) indexes different iterations. The necessary and sufficient condition for convergence is \( \rho(A) < 1 \), where \( \rho(A) \) is the spectral radius of matrix \( A \), defined as \( \rho(A) = \max_i |\lambda_i| \), \( \lambda_i \) being an eigenvalue of \( A \). The speed at which \( x_k \) approaches \( x^* \), called the convergence rate, is governed by the spectral radius \( \rho(A) \) since the error vector \( e_k \) is such that:

Performance of Asynchronous Linear Iterations with Random Delays

Adrian C. Moga 
Michel Dubois
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562
{moga,dubois}@paris.usc.edu

* This research has been funded by the National Science Foundation Grant No. CCR-9222734.
When the norm used for the error vector is Euclidean and when \( A \) is normal, the corresponding matrix norm is spectral and \( \|A^k\| = \rho(A^k) = \rho^k(A) \), thus the error vector asymptotically approaches 0 at the same rate as \( \rho^k(A) \).

**Asynchronous Iterations:** under deterministic assumptions, a sufficient condition for convergence is \( \rho(|A|) < 1 \) where \( |A| \) is the matrix with elements \( |a_{ij}|\). Convergence conditions results for models with stochastic delays will be given in Section 4.

### 2.2. Oscillations and Damping Effects

Error oscillations in synchronous linear iterations and their asynchronous damping were first reported by Bull and Freeman [5]. They use the cosine of two consecutive error vectors \( C(k) \) to measure the oscillations of the sequence of error vectors in the synchronous iteration. Strong oscillations are observed when \( \lambda_1 \), the dominant eigenvalue of \( A \), is negative and \( C(k) \rightarrow -1 \). The effect of asynchronism on Jacobi iterations is analyzed for several linear systems, especially chosen to exhibit various oscillation degrees. Data collected during actual runs shows consistent asynchronous slowdown when \( \lambda_1 > 0 \), consistent asynchronous speedup when \( \lambda_1 < 0 \) (more prominent as \( \lambda_1 \) is closer to -1) and inconclusive results for complex \( \lambda_1 \). The asynchronous speedup effect was attributed to the damping of the oscillatory behavior without any further analysis. Similar observations were made in [11].

**Figure 1: Error vector oscillation when \( \lambda_1 < 0 \).**

The error vector can be expanded as

\[
\varepsilon_k = A^k \varepsilon_0 = \lambda_1^k \chi_1 + \lambda_2^k \chi_2 + \ldots
\]

When \( \lambda_1 \) is real and \( \lambda_1 \neq \lambda_2 \), \( \varepsilon_k \) asymptotically coaxes with \( \chi_1 \), the associated eigenvector of \( \lambda_1 \). If \( \lambda_1 > 0 \) the error vectors stay in the direction of \( \chi_1 \), whereas when \( \lambda_1 < 0 \), they switch direction at each iteration (Figure 1). When \( \lambda_1 = \rho e^{i\alpha} \) is complex, \( \chi_2 = \lambda_2^k \chi_1 \), and \( ||\chi_1|| > ||\chi_2|| \). \( \chi_1 = a + bi \) and \( \chi_2 = a - bi \), with \( a \) and \( b \) real vectors. For sufficiently large \( k \),

\[
\varepsilon_k = \lambda_1^k \chi_1 + \lambda_2^k \chi_2 = 2^{k} \left( \cos(k\alpha) a - \sin(k\alpha) b \right)
\]

showing that \( \varepsilon_k \) belongs to the subspace spanned by \( a \) and \( b \). When there is an integer \( n \) such that \( \alpha = 2\pi/n \), \( \varepsilon_k \) periodically occupies the same positions in space (Figure 2). In general, this shows that the \( \varepsilon_k \) of the synchronous iteration become asymptotically “trapped” into subspaces, commonly of dimension one or two, and the convergence takes place at a geometric rate, which is slow when \( \rho(A) \approx 1 \).

**Figure 2: Error vector oscillation when \( \lambda_1 = \rho e^{i\pi/2} \).**

By introducing non-linear recombination operations, the asynchronous sequence may break away from the restrictive subspaces and, sometimes, converge very fast. When the synchronous error vectors are all contained in the same “cone” of Figure 1 or 2, no combination of vector components under asynchronism can take the error vector out of the cone. By contrast, super-linear asynchronous convergence is observed when \( \lambda_1 \) is negative because of the randomness in combining iterate components from consecutive iterations (i.e. approaching the solution \( x^* \)) from opposing cones in Figure 1). This leads to the damping of the oscillation by mutual cancelation of positive and negative error components. Similar damping effects occur for complex \( \lambda_1 \) in the presence of strong synchronous oscillations when \( \alpha = 2\pi/n \).

The damping of oscillations is most prominent when the error vectors for the synchronous iteration, collected during a time window equal to the maximum asynchronous delay, have a sum of approximately zero, assuming a uniform distribution of delays. In particular, when \( \lambda_1 = -1 \), two consecutive error vectors may cancel each other. When \( \alpha = 2\pi/n \), asynchronous damping is predominant when the maximum asynchronous delay is a multiple of \( n \). The chances of performing a strong cancellation, in general, increase when the odds of combining components from opposite cones are high, such as when there are fewer cones to pick from during the maximum delay time window (e.g. just two, when \( \alpha = \pi \)) and when consecutive vectors have very similar magnitudes (i.e., \( \rho(A) = 1 \)). In conclusion, asynchronous speedup is highest when \( \lambda_1 \) is close to -1. From our experiments, this condition appears to be necessary to achieve reasonable speedup through oscillation damping. It is, however, not sufficient. A “close coupling” between equations is also desirable, in order to have many possibilities to cancel the oscillations.
2.3. Performance Metrics

Our primary performance metric is the asynchronous speedup, i.e., the ratio between the numbers of synchronous and asynchronous iterations needed to converge:

\[
S_p = \frac{N_{\text{sync}}(A)}{N_{\text{async}}(A)}
\]

The converge criterion is based on the Euclidean distance between the current iterate and the actual solution \(x^*\). For synchronous methods, the evolution of the iteration vector \(x\) from \(x_0\) to the final value is unique and \(N_{\text{it}}\) is obtained in a single simulation run. \(N_{\text{it}}\) can also be estimated analytically for the Jacobi method [8], \(ζ\) being the error reduction:

\[
N_{\text{it}} = \langle \log ζ \rangle / (\log ρ(A))
\]

In asynchronous methods, the trajectory of the iteration vector and the number of iterations \(N_{\text{it}}\) depend on the execution environment. The expected trajectory of the iterate vector is estimated by generating multiple random trajectories based on some stochastic distribution and the number of iterations is taken as the average of the number of iterations in each trajectory. Analytical estimates of the average number of iterations can also be obtained by a formula similar to (5), where \(A\) is replaced by an augmented matrix taking into account the distribution of possible trajectories.

Given a finite set \(\mathcal{A}\) of matrices, we define the average number of iterations until convergence over \(\mathcal{A}\) as:

\[
\mathbb{E}[N_{\text{it}}(\mathcal{A})] = \sum_{A \in \mathcal{A}} N_{\text{it}}(A) W(A)
\]

\(W(A)\) is a weight associated with matrix \(A\).

The set \(\mathcal{A}\) in this paper was derived by discretizing the space of 2x2 matrices of real numbers for which \(\|A\|_{∞} < 1\). A total number of 266,256 matrices was obtained using a non-uniform discretization grid. More sample points were selected at the edges of the domain because the behavior of asynchronous algorithms changes faster in these areas.

3. Simulation Model

Given the iteration matrix \(A\), the linear iteration

\[
X_{k+1} = \begin{bmatrix} x_{k+1}^0 \\ x_{k+1}^1 \end{bmatrix} = \begin{bmatrix} 0^0 x_k^0 + a_{01} x_k^1 \\ a_{00} x_k^0 + a_{11} x_k^1 \end{bmatrix}
\]

is assigned for execution on a 2-processor system, each processor being in charge of one iteration variable. Processors repeatedly fetch the current value of both variables, compute the value of the assigned component and update it, until the detection of the termination condition: \(\|x_k\| < ε = 10^{-4}\). The computation time of processor \(P_i\) in iteration \(k\) is \(t_{\text{it}}(k) = t_{\text{const}} + t_{\text{rand}}(k)\). \(t_{\text{rand}}\) is a random fluctuation generated independently for each processor from a uniform distribution in the interval \([0, \text{fluct}_{\text{max}}]\). In the asynchronous case, the simulation is repeated 100 times with different sequences of fluctuation components.

Figure 3: 3-D visualization of the asynchronous speedup for \(t_{\text{const}} >> \text{fluct}_{\text{max}}\)

Figure 3 displays the asynchronous speedup for a set of matrices with fixed \(a_{10}, a_{11}\). Speedup is slightly less than 1 for most of the points in the plot, but it soars to tens and hundreds (visible on a finer grid), generating a thin peak, culminating around \(a_{00} = 0.85\) and \(a_{01} = 0.125\). Towards the edge of the convergence domain, as \(ρ(A) → 1\), the value of the asynchronous speedup becomes unbounded.

We now examine closer why the asynchronous iteration converges much faster than the synchronous iteration for \(A = (-0.83, 0.16); (0.974, -0.025)\) and \(\text{fluct}_{\text{max}} = 2\). This case exhibits an asynchronous speedup of about 60. The trajectory of the Jacobi synchronous iteration in Figure 4 shows very slow convergence in an oscillatory regime. The spectral radius \(ρ(A) \approx -0.991\), explaining the slow convergence, and the dominant eigenvalue for \(A\) is 0.991, causing the oscillations, as explained in Section 2.

Figure 4: Trajectories of the iteration vector.

The asynchronous trajectory is a closing spiral, the computation following a Gauss-Seidel evolution for the first six steps: updates of \(x_1\) follow updates of \(x_0\). In this regime the spectral radius for the modified iteration matrix is 0.666, considerably smaller than \(ρ(A)\), which explains the faster convergence. At times, convergence accelerates...
beyond the Gauss-Seidel effect. Several such occurrences further cut down on the total number of iterations, leading to even higher speedups. But hazard is not always so generous. Sometimes, the steady Gauss-Seidel evolution is slowed down by the occurrence of such changes in the sequence of computations. Over large number of trials, it all events out such that the speedup is largely attributable to the Gauss-Seidel effect. A reverse effect can also be observed in some cases, when the spectral radius of the iteration matrix corresponding to the approximate Gauss-Seidel asynchronous execution is larger than \( \rho(A) \).

The height of the peak in Figure 3 depends on the value of \( \text{fluct}_{\text{max}} \) relative to \( t_{\text{const}} \). It is largest for very small values of the fluctuations, in “nearly synchronous” conditions. As the fluctuation grows, the peak value of the asynchronous speedup decreases, eventually stabilizing when \( \text{fluct}_{\text{max}} = t_{\text{const}} \), as shown in Figure 5. It appears that small fluctuations favor a stricter alternation of the computation of the two variables in a Gauss-Seidel sequence whereas, for larger fluctuations, some benefits of this effect are lost because of frequent changes in the order of evaluation.

![Figure 5: Maximum speedup on the sectioning grid of Figure 2 as a function of \( \text{fluct}_{\text{max}} \).](image)

Figure 6 plots the average asynchronous speedup for different magnitudes of the fluctuation, showing that the number of iterations until convergence is, on the average, slightly smaller in the synchronous iteration. The speedup is also not very sensitive to the magnitude of the fluctuation. Overall, the synchronous iteration performs better.

4. Analytical Model with Stochastic Delays

In [2] a model of asynchronous linear iterations with stochastic delays was developed and sufficient conditions for convergence were derived. In the special case when delays are independent, it is shown that a sufficient conditions for convergence in the mean and in the second moment is:

\[
\rho(\sum_{m=1}^{l} C_m p_m) < 1
\]

\( C_m \) is a matrix describing a possible recurrence relation in the augmented space. If \( B_{ij} \) is the maximum delay to propagate a message from processor \( i \) to processor \( j \), then there is one \( C_m \) matrix (with probability \( p_m \)) for each of the \( l=\prod_{j=1}^{l} B_{ij}+1 \) cases obtained by combining possible delays for the use of \( x_j \) in the computation of \( x_i \). The asynchronous speedup for the asynchronous model with independent stochastic delays is:

\[
S = \frac{\log(\rho(\sum_{m=1}^{l} C_m p_m))}{\log(\rho(A))}
\]

Consider the case where the probabilities that a value is communicated to processor 1 (respectively 2) with a delay of zero or one are \( p_1 \) and \( 1-p_1 \) (respectively \( p_2 \) and \( 1-p_2 \)). Taking the expected values of both sides of relation (7), we obtain the following relations for the expected asynchronous trajectory:

\[
\begin{align*}
x^n_{10} &= a_{00} x^n_{00} + a_{01} (p_1^n x^n_{11} + (1-p_1^n) x^n_{01}) \\
x^n_{11} &= a_{11} x^n_{10} + a_{10} (p_2^n x^n_{00} + (1-p_2^n) x^n_{01})
\end{align*}
\]

Using the space augmentation technique in [2], the following recurrence relation, involving \( A_{\text{aug}} \), is obtained:

\[
\begin{bmatrix}
x^n_{10} \\ x^n_{00} \\ x^n_{11} \\ x^n_{01}
\end{bmatrix} =
\begin{bmatrix}
a_{00} & 0 & p_1 a_{01} (1-p_1) a_{01} & 0 \\
1 & 0 & 0 & 0 \\
p_2 a_{10} (1-p_2) a_{10} & 0 & a_{11} & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x^n_{10} \\ x^n_{00} \\ x^n_{11} \\ x^n_{01}
\end{bmatrix}
\]

For given \( p_1, p_2 \), the asynchronous speedup is given by:

\[
\frac{\log(\rho(A_{\text{aug}}))}{\log(\rho(A))}
\]

To see the effect of the delay probabilities, we have computed the average asynchronous speedup over all matrices in the set and for each possible combination of \( p_1 \) and \( p_2 \).

![Figure 6: Average asynchronous speedups as a function of \( \text{fluct}_{\text{max}} \).](image)
between 0 and 1, with a resolution of 0.1. The results are plotted in Figure 7. Small amounts of asynchronism can produce slight improvements of up to 10%, over the synchronous execution. As the older iterate values are used (i.e., \( p_1 \) and \( p_2 \) are close to zero), the performance quickly degrades by as much as 30%. When \( p_1=p_2=0.5 \), the expected speedup is 0.9895, matching the results obtained through simulation from Section 3.

Figure 7: Expected asynchronous speedup as a function of the delay probabilities.

The maximum asynchronous speedup for each iteration matrix \( A \) is obtained by selecting probabilities \( p_1 \) and \( p_2 \) such that the spectral radius \( \rho(A_{\text{async}}) \) is minimized. We then compute the best-case average speedup by taking the mean of the maximum asynchronous speedups over all matrices. The value of the maximum speedup is strongly correlated to the spectral radius of \( A \) and reasonable maximum speedup values are only observed for \( \rho(A) \geq 0.9 \). Figure 8 shows that only a small fraction of matrices yield maximum asynchronous speedups of at least one order of magnitude. The average of this best case asynchronous speedup, computed over the entire set of matrices, is 1.38.

Figure 8: Maximum speedup histogram.

5. Conclusions

We have performed a systematic investigation of the effects of asynchronism on the convergence rate of linear iterations of size two. Our results show that asynchronous linear iterations may occasionally yield a very rapid convergence attributed to oscillations present in the synchronous execution, especially when the iteration matrix has a dominant eigenvalue in the neighborhood of -1; these oscillations are quickly damped in the presence of asynchronism. However, such instances are not very frequent and the average speedup over all matrices is less than one. On the other hand, the models show that the asynchronous iteration does not perform much worse than its synchronous counterpart even in the worst case. Thus, overall, asynchronous iterations might still consistently outperform synchronous methods once the overhead of synchronization primitives is included.

References