{youyang, demmel}@eecs.berkeley.edu {kentcz, lsong, richie}@gatech.edu

Appendix of Communication-Avoiding SVM

by Yang You, James Demmel, Kent Czechowski, Le Song, Richard Vuduc

1 Efficiency of CA-SVM

In the following, we use n for simplicity to refer to the problem size, whereas our paper probably uses $m \times n$ for number of samples $\times$ dimension. Like our paper, we use P to refer to the number of nodes. To be more precise, let $t(n; P)$ be the per-iteration time, which is a function of n and P; and let $i(n; P)$ be the number of iterations, also possibly a function of n and P. For Dis-SMO, we observe $i(n; P) = \Theta(n)$, that is, there is no actual dependence on P. Then, the total time should really be

$$T(n; P) = i(n; P) * (\Theta(1) + t(n; P))$$

The reason for the $\Theta(1)$ term is to capture the fact that even if $t(n; P)$ is very small, we still have to execute the iterations sequentially. That is, executing the iterations may be on the critical path, even if each iteration is so efficient that $t(n; P)$ is close to zero. Thus, the efficiency becomes

$$E(n; P) = \frac{i(n; 1) \times t(n; 1)}{P \times i(n; P) \times (\Theta(1) + t(n; P))}$$

For Dis-SMO, $i(n; 1) = i(n; P)$, which means

$$E(n; P) = \frac{t(n; 1)}{P \times (\Theta(1) + t(n; P))}$$

If the per-iteration time scales perfectly — meaning $t(n; P) = t(n; 1)/P$ — the efficiency should be

$$E(n; P) = \frac{t(n; P)}{\Theta(1) + t(n; P)}$$

The efficiency of SMO can be close to 1 if $\Theta(1)$ is small.

For CA-SVM, each node is actually an independent SVM. Thus we can get that $i(n; P) = \Theta(n/P)$ because each node only trains n/P samples. In other words, each node is a SMO problem with n/P samples.

$$E(n; P) = \frac{i(n; 1) \times t(n; 1)}{P \times i(n; P) \times (\Theta(1) + t(n; P))}$$

Therefore, $i(n; 1)$ is close to $P * i(n; P)$, which means

$$E(n; P) = \frac{t(n; 1)}{\Theta(1) + t(n; P)}$$

On the other hand $t(n; 1)$ is close to $P \times t(n; P)$ because each node only has n/P samples. In this way, we get

$$E(n; P) = \frac{P \times t(n; P)}{\Theta(1) + t(n; P)}$$

This means the efficiency of CA-SVM is close to P if $\Theta(1)$ is small.

Usually, we expect efficiency to lie between 0 and 1. The way we set this up is perhaps not quite right – the sequential baseline should be the best sequential baseline, not the naive (plain SMO) one. If we execute CA-SVM sequentially by simulating P nodes with only 1 node, then the sequential time would be

$$P * (i(n; 1)/P * t(n; 1)/P) = i(n; 1) * t(n; 1)/P$$

Table 1: Terms for Performance Modelling

m; n; P	# samples; # features per sample; # nodes or processes
T_1; T_p	serial run time; parallel run time
t_s; t_w	startup time for communication; per-word transfer time
S_k	# SVs in k_{th} Cascade layer, $S_1 = m$
I_k	maximal # iters of all nodes in k_{th} Cascade layer
P_k	# processes in k_{th} Cascade layer, $P_k = \frac{1}{2}P_{k-1}, P_1 = P$
W; T_o	problem size; parallel overhead ($T_o = PT_p - W$)
s; I; k	# SVs; # SVM iters; # k-means iters

So then $E(n; P)$ would approach 1 rather than P. Put another way, this means that SMO is not a good algorithm, even in the sequential case. That is, we can beat SMO by running CA-SVM to simulate P nodes using only 1 node.

2 Performance Modeling

This section we give the detailed modeling for these approaches, the terms used in this section are in Table 1.

Dis-SMO: We use linear kernel for simplicity. The serial run time of single SMO iteration is $2mn$. The run time of other kernel is longer. First of all, we model the parallel run time of single SMO iteration.

- Broadcast X_{high} and X_{low}
  - $2(t_s + nt_w)logP$
- All Nodes: Computation and Local Reduce
  - $\frac{2mn}{P} + \frac{4m}{P}$
- Gather local $i_{high}, i_{low}, b_{high}, b_{low}$ to master node
  - $4(t_s logP + t_w P(P-1))$
- Master Node: get global $i_{high}, i_{low}, b_{high}, b_{low}$
  - $2P + n$
- Broadcast $i_{high}, i_{low}, b_{high}, b_{low}$ and other 4 related terms
  - $8(t_s + t_w)logP$
- Parallel Run Time
  - $T_p = 2(t_s + nt_w)logP + \frac{2mn}{P} + \frac{4m}{P} + 4(t_s logP + t_w P(P-1)) + 2P + n + 8(t_s + t_w)logP$
  - $= 14logPt_s + [(2n+8)logP + 4P(P-1)]t_w + \frac{2m(n+2)}{P} + 2P + n$

.

- Parallel Overhead: $T_o = PT_p - W$
  - $= 14PlogPt_s + [(2n+8)PlogP + 4P^2(P-1)]t_w + 2m(n+2) + 2P^2 + nP - 2mn$
  - $= 14PlogPt_s + [(2n+8)PlogP + 4P^2(P-1)]t_w + 4m + 2P^2 + nP$

Next, we do the scaling analysis based on iso-efficiency function.

- Iso efficiency function
  - $W = KT_o$, $K = E/(1-E)$
  - E is the desired efficiency
- $W = KT_o = 14KPlogPt_s + [(2n+8)PlogP + 4P^2(P-1)]Kt_w + (4m + 2P^2 + nP)K$
- For communication startup time t_s
  - $W = \Theta(PlogP)$
  - 1-D and 2-D Mat-Vec Mul: $W = \Theta(PlogP)$
- For communication transfer time t_w, **Bad Scaling**
  - $W = \Theta(max(nPlogP, P^3)) \geq \Theta(P^3) \Rightarrow \Omega(P^3)$
  - 1-D Mat-Vec Mul: $W = \Omega(P^2)$, 2-D Mat-Vec Mul: $\Omega(P)$
- For computation time, **Bad Scaling**
  - $W = \Theta(max(P^2, nP)) \geq \Theta(P^2) \Rightarrow \Omega(P^2)$
  - 1-D and 2-D Mat-Vec Mul: $W = \Omega(1)$

Cascade: In this section we do the Performance Modeling of Cascade. It is for one pass, not for one iteration. $T_1 = I \times 2mn = 2Imn$. The following is the modeling of parallel run time:

- Computation of k_{th} layer
  - $2\frac{S_{k-1}}{P_k} nI_k$ $(S_0 = m)$
- Communication of k_{th} layer: $\frac{P_k}{2}$ two-node gather operations
  - $(log2)t_s + n\frac{2S_k}{P_k}t_w = t_s + n\frac{2S_k}{P_k}t_w$ **(in parallel)**
  - $k \in 1, 2, ..., (logP - 1)$
- Communication of the last layer: scatter SVs to all the nodes
  - $(logP)t_s + S_{logP}(P-1)t_w = (logP)t_s + S_{logP}Pt_w$
  - S_{logP} is the number of SVs in the last layer
- Run time T_p
  - $2n\sum_{k=1}^{logP} \frac{I_k S_{k-1}}{P_k} + \sum_{k=1}^{logP-1}(t_s + n\frac{2S_k}{P_k}t_w) + (logP)t_s + S_{logP}Pt_w$
  - $= 2n\sum_{k=1}^{logP} \frac{I_k S_{k-1}}{\frac{1}{2^{k-1}}P} + (logP - 1)t_s + \sum_{k=1}^{logP-1}(n\frac{2S_k}{\frac{P}{2^{k-1}}}t_w) + (logP)t_s + S_{logP}Pt_w$
  - $= \frac{2n}{P}\sum_{k=1}^{logP} I_k S_{k-1}2^{k-1} + (logP)t_s + \sum_{k=1}^{logP-1}(n\frac{2^k S_k}{P}t_w) + (logP)t_s + S_{logP}Pt_w$
  - $= \frac{n}{P}\sum_{k=1}^{logP} I_k S_{k-1}2^k + (2logP)t_s + [(\sum_{k=1}^{logP-1}\frac{n2^k S_k}{P}) + PS_{logP}]t_w$
- Parallel Overhead: $T_o = PT_p - W$
  - $= n\sum_{k=1}^{logP} I_k S_{k-1}2^k + (2PlogP)t_s + [(\sum_{k=1}^{logP-1} n2^k S_k) + P^2 S_{logP}]t_w - 2Imn$
  - $= n\sum_{k=1}^{logP} I_k S_{k-1}2^k - 2Imn + (2PlogP)t_s + [(\sum_{k=1}^{logP-1} n2^k S_k) + P^2 S_{logP}]t_w$
  - $= n(\sum_{k=1}^{logP} I_k S_{k-1}2^k - 2Im) + (2PlogP)t_s + [(\sum_{k=1}^{logP-1} n2^k S_k) + P^2 S_{logP}]t_w$

The scaling analysis for Cascade SVM are the following:

- Iso efficiency function
  - $W = KT_o$, $K = E/(1-E)$
  - E is the desired efficiency
- $W = KT_o = nK(\sum_{k=1}^{logP} I_k S_{k-1} 2^k - 2Im) + (2KPlogP)t_s + [(\sum_{k=1}^{logP-1} n2^k S_k) + P^2 S_{logP}]Kt_w$
- For communication startup time t_s
  - $W = \Theta(PlogP)$
  - Distributed SMO: $W = \Theta(PlogP)$
- For communication transfer time t_w
  - $W = \Theta((\sum_{k=1}^{logP-1} n2^k S_k) + P^2 S_{logP})$
  - Distributed SMO: $W = \Theta(max(nPlogP, P^3)) \geq \Theta(P^3) \Rightarrow \Omega(P^3)$
- For computation time
  - $W = \Theta(n(\sum_{k=1}^{logP} I_k S_{k-1} 2^k - 2Im))$
  - Distributed SMO: $W = \Theta(max(P^2, nP)) \geq \Theta(P^2) \Rightarrow \Omega(P^2)$

3 Modeling of Communication Volume

In this section, we model the communication amount for these six different approaches. int denotes integer and float denotes floating point. We normalize that the size of int is 1 and the size of float is 2.

Dis-SMO:
Bcast:
2*p int + 6*p float+2I*p float+2I*p int + 6I*p float+2*p int + 1*pm float
=2*p + 6*2p + 2I*2p + 2I*p + 6I*2p + 2*p + 1*2pm
=2p + 12p + 4Ip + 2Ip + 12Ip + 2p + 2pm
=16p+18Ip+2pm
Scatter:
1*m float + 1*p int + 1*nnz float + 1*nnz int +1* m int
= 1*2m + 1*p + 1*mn + 1*2mn + 1*mn + 1*m
= 3m + p + 4mn
Gather:
I*4p float
=I*8p=8Ip
Total:
16p+18Ip+2pm+3m+p+4mn+8Ip
=17p+26Ip+2pm+3m+4mn
=26Ip+2pm+4mn

Cascade:
Bcast:
2*p int
=2p
Scatter:
1*p int + 1*nnz float + 1*nnz int + 1*m float + 1*m int
=1*p+1*2mn+1*mn+1*2m+1*m
=p+3mn+3m
Send/Recv:
We first get the communication amount of the logp level:
(2*2 int + svnnz/p float + svnnz/p int + sv/p float + sv/p float + sv/p int)*(p/2)

=(2*2 int + svnnz/p float + svnnz/p int + 2*sv/p float + sv/p int)*(p/2)

In the above equation, for 2*2 int, the number of operations is decreasing at a factor of 2. Thus, 2*2(p/2 + p/4 + p/8 + ... + 1) = 2p*2

For the rest, we can only get the upper bound since we can not predict the number support vectors on each layer, which means each layer has the same number of support vectors (actually the number of svs is decreasing layer-by-layer). So totally $log(p)$ layers have communication.

Sum of the Send/Recv:
2p*2 int + (svnnz/p float + svnnz/p int + 2*sv/p float + sv/p int)*p
=2p*2 int + O(p*(svnnz/p float + svnnz/p int + 2*sv/p float + sv/p int))
=2p*2 int + O(p*(nnz/p float + nnz/p int + 2*s/p float + s/p int))
=2p*2 + O(p*(2sn/p + sn/p + 2*2s/p + s/p))
=4p+O(3sn + 5s)
Total:
2p+p+3mn+3m+4p+O(3sn+5s)
=O(7p+3mn+3m+3sn+5s)
=O(3mn+3m+3sn)

DC-SVM:
Bcast:
2*p int + 1*pn float
=2*p + 1*2pn
=2p+2pn
Scatter:
1*p int + 1*nnz float + 1*nnz int + 1*m float + 1*m int + 1*p int + 1*nnz float + 1*nnz int + 1*m int + 1*m float
=1*p + 1*2mn + 1*mn + 1*2m + 1*m + 1*p + 1*2mn + 1*mn + 1*m + 1*2m
=p + 2mn + mn + 2m + m + p + 2mn + mn + m + 2m
=2p + 6mn + 6m
Send/Recv:
2p*2+(p*(nnz/p float + nnz/p int + 2*m/p float + m/p int))
=2p*2+(p*(2nnz/p + nnz/p +4m/p + m/p))
=4p+(2nnz + nnz +4m + m)
=4p+(3nnz +5m)
=(3nnz +5m+4p)
MPI-Allreduce:
k*pn float + k*p int + k*1 float
=k*2pn + k*p + k*2
=2kpn + kp + 2k
MPI-Gather:
1*m int=1*m=m
Total:
2p+2pn + 2p + 6mn + 6m+(3nnz +5m+4p)+2kpn + kp + 2k+m
=2p+2pn + 2p + 6mn + 6m+3nnz +5m+4p+2kpn + kp + 2k+m
=8p+2pn+9mn+12m+2kpn+kp+2k
=9mn+12m+2kpn

DC-Filter:
Bcast:
2*p int + 1*pn float
=2p+2pn
Scatter:
1*p int + 1*nnz float + 1*nnz int + 1*m float + 1*m int + 1*p int + 1*nnz float + 1*nnz int + 1*m int + 1*m float
=p + 2nnz + nnz + 2m + m + p + 2nnz + nnz + m + 2m
=6nnz + 6m + 2p

Send/Recv:
2p*2 + O(p*(snnz/p float + snnz/p int + 2*s/p float + s/p int))
=4p + O(p*(3snnz/p + 5s/p))
=4p + O(3snnz + 5s)
=O(4p+3sn+5s)
MPI-Allreduce:
k*pn float + k*p int + k*1 float
=2kpn + kp + 2k
MPI-Gather:
1*m int=m
Total:
2p+2pn+6nnz+6m+2p+O(4p+3sn+5s)+2kpn+kp+2k+m
=4p+2pn+6mn+7m+O(4p+3sn+5s)+2kpn+kp+2k
=O(8p+2pn+6mn+7m+3sn+5s+2kpn+kp+2k)
=O(6mn+7m+2kpn)

CP-SVM:
Bcast:
2*p int + 1*pn float
=2p+2pn
Scatter:
1*p int + 1*nnz float + 1*nnz int + 1*m float + 1*m int + 1*p int + 1*nnz float + 1*nnz int + 1*m int + 1*m float
=p+2nnz+nnz+2m+m+p+2nnz+nnz+m+2m
=2p+6nnz+6m
=2p+6mn+6m
MPI-Allreduce:
k*pn float + k*p int + k*1 float
=2kpn+kp+2k
MPI-Gather:
1*m int=m
Total:
2p+2pn+2p+6mn+6m+2kpn+kp+2k+m
=4p+2pn+6mn+7m+2kpn+kp+2k
=6mn+7m+2kpn

CA-SVM: For CA-SVM, the communication amount is 0.

Table 2: Modeling of Communication Volume

Method	Formula	Prediction	Test
Dis-SMO	$\Theta(26Ip + 2pm + 4mn)$	36MB	34MB
Cascade	$O(3mn + 3m + 3sn)$	8.4MB	8.4MB
DC-SVM	$\Theta(9mn + 12m + 2kpn)$	24MB	29MB
DC-Filter	$O(6mn + 7m + 3sn + 2kpn)$	16.2MB	18MB
CP-SVM	$\Theta(6mn + 7m + 2kpn)$	15.6MB	17MB
CA-SVM	0	0MB	0MB