1 Introduction

- the DBLAS Library is a portable || BLAS, highly tuned for the AP1000/AP+
- to transport the DBLAS to the Fujitsu AP3000 & VPP-300:
  - which have very different cell architectures:
    - AP3000: RISC scalar processors (U170's) with a deep memory hierarchy
    - VPP-300: vector processors with high bandwidth interleaved memory
  - communication networks on both unlike that on the AP1000/AP+:
    - high communication costs relative to floating point speed
    - row or column broadcasts must be simulated by point-to-point messages
    - no stride communication

  requires special care to ensure high performance, needing:

  - tuning parameters to select optimal block sizes, local matrix storage, etc.
- note: even if an efficient BLAS is available, cell computation cannot be ignored:
  - eg. handling the local portions of Δ matrices on a $P \times Q$ grid (eg. in || symmetric rank-k updates, the Parallel BLAS must partition the local Δ matrix)

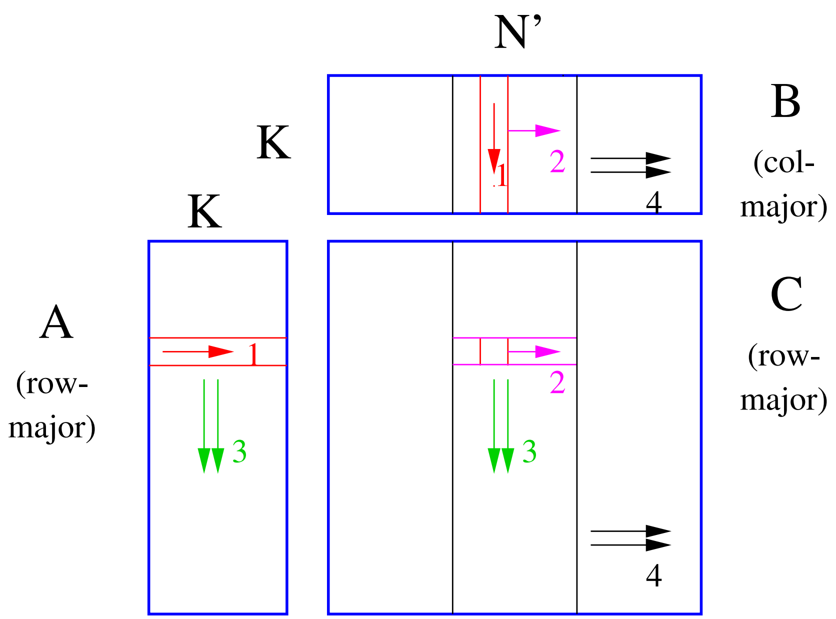
2 Memory Access Pattern Issues: local Matrix Transpose

- not available in the serial BLAS; for the DBLAS, required for:
  - packing of the 'panels' (eg. in || matrix factorization)
    - minimizes cache / TLB misses in the panel formation
    - also for spread and reduce communications of the panel (unit stride along the length of the panel is optimal)
  - to get the optimal matrix storage scheme for A, B for local matrix multiply
- for $B \leftarrow A^T$, where B is $M \times N$ and $MN > |\text{TLB}|\ |\text{Page}|$:
  - if N is optimal for the *innermost iteration*, partition A, B into $M \times N'$ blocks, where $N' = |\text{TLB}|/2$
  - this ensures there will be least M (N') accesses to B (A) per TLB miss
- performance on an UltraSPARC U170:
  -

	partitioning (N'=32)	unoptimized
`LdB` $= N$, $2048 \leq N = M \leq 4000$	85–100 MB/s	30 MB/s
`LdA` $= M$, $2048 \leq M \leq 4000$, $N = 128$	85–95 MB/s	7–23 MB/s

3 Memory Access Pattern Issues: local Matrix Multiply

- consider the multiply $C + = AB$, where C is $N \times N$ and A is $N \times K$ on a U170
- inner loop kernel updates (a 4×4 sub-matrix of) C using $4{\times}K$ ($K{\times}4$) strip of A (B)
  - uses cache lookahead to minimize top-level cache misses
  - achieves 300–320 MFLOPs for $K \approx 128$ and A & B in top-level cache
- for large N, the following alg. ensures unit stride along the 2 innermost iterations:



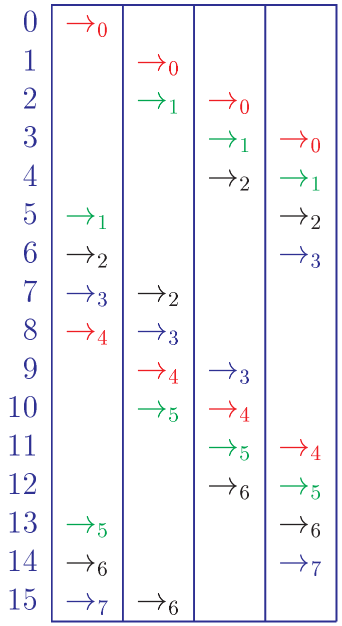
- partition B st. $KN' \approx \min(|\text{level-2 cache}|, |\text{TLB}| * |\text{page}|/2)$
  - ie. $K \times N'$ sub-block of B takes no more than 1/2 the TLB entries
- on UltraSPARC I, optimal $K = 88, N' = 256$, sustained 260–280 MFLOPs for $512 \leq N < 6144$

4 The U170 BLAS: Level 1 & 2

- general construction principles: achieve compact code size with high speed:
  - issue loads 8–10 cycles before 1st use (achieves (top-level) cache lookahead)
  - perform minimal innermost loop unrolling to reduce the startup costs
- performance on various computations (with unit stride):
function	effect	U170 BLAS		unrolling	Sun Perf. Lib 1.2		theor. max.
`dgemv()`	$y \leftarrow Ax$	240	(256)	8×2	214	(213)	330
`dger()`	$A \leftarrow A + xy^T$	140	(100)	8×1	121	(100)	165
`dnrm2()`	$(x \cdot x)^{1/2}$	250	(250)	1×8	50	(45)	330
`ddot()`	$x \cdot y$	139	(139)	1×4	54	(45)	165
`daxpy()`	$y \leftarrow \alpha x + y$	97	(78)	1×4	74	(80)	126
`dscal()`	$y \leftarrow \alpha y$	80	(66)	1×4	65	(48)	85
`dcopy()`	$y \leftarrow x$	64	(65)	1×8	78	(81)	85
`dswap()`	$y \leftrightarrow x$	35	(32)	1×4	40	(29)	43
`idamax()`	i s.t. $\|x_i\| \geq \max_{j=0}^{n-1} \|x_j\|$	78	(83)	1×5	40	(29)	85
  - A – $n \times n$; x, y – $1 \times n$. Speeds in MFLOPs.
  - n chosen s.t. data size = 16 KB (128 KB)

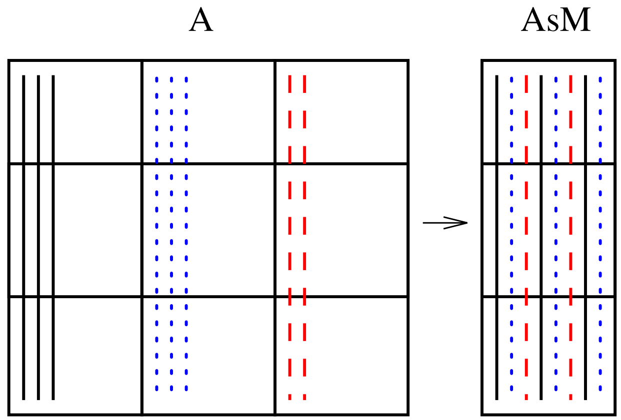
5 Communication Enhancements for the DBLAS

- pipelined broadcasts: minimize startups!
  - reduces relative cost κ_{bc} from $\lg_2(Q)$ (for tree broadcasts) to 2 across a grid of width Q



  - requires the fire-and-forget semantics
  - combining with lookahead techniques (eg. in matrix factorization) reduces $\kappa_{bc} \approx 1$

- efficient row/column spreads



  - again, for AP3000 & VPP, can only use point-to-point messages
  - for K columns, achieve by $Q-1$ ring-shifts of width $\approx \lceil \frac{K}{Q} \rceil$
  - relative cost is $\kappa_{bc} \approx \frac{Q-1}{Q}$
  - can use similar techniques for reduce operations

6 Further DBLAS Communication/Computations Issues

- reduce commun. startups on $B \leftarrow A^{-1}B$ operations, where A is Δ & small, by:
  - spreading A before the computation (can also do to B for $B \leftarrow AB$)
  - can pipeline the broadcasts of the rows of B
    - relative cost is effectively $\kappa_{bc} = \frac{r+1}{r}$, where r is the storage block size
    - by blocking these broadcasts, relative cost reduces to $\kappa_{bc} = \frac{2}{r}$
- DBLAS interface for External (eg. vendor-supplied) BLAS
  - ideally must be insensitive / avoid any performance ideosyncrasies of this BLAS
  - considerations include:
    - avoid potential software overheads by calling Level 1 & 2 instead of Level 3
    - decide whether it is optimal to 'prescale' matrix input operands
    - try to use the BLAS's 'preferred' matrix storage scheme (problematic)
- use a tuning parameter to indicate if a daxpy()-based matrix multiply is optimal (eg. on the VPP)
  - DBLAS algorithms must ensure unit stride for this case too