Interaction overhead

① 줄이기 위해 고려할 요소

- data locality ↑ … Communication for data 횟수 ↓
- graph partitioning … crossing edge ↓ … 구조적 문제 해결하기
- Communication frequency ↓ … 한번에 chunk/batch communicate 해서 frequency 줄이기
- Contention/hot-spot ↓ … bottleneck 피하기 위해 de-centralized dynamic mapping 사용

② 방법

- overlapping communication-computation
  - single-thread : interrupt 사용해서 CPU idle 줄이기
    prefetch 이용해서 latency 줄이기
  - multi-thread : communication을 other computation과 overlapping

- replicate data
  - redundancy를 조금 희생해서 communication 줄이기
  - false-sharing도 줄일 수 있음 (local copy 유지)

- overlapping communications
  - stepwise : 1level task의 대해서 pipeline
  - naive . 그냥 뿌려주기
  - pipelined naive : 전체 pipeline

Minimizing Interaction Overhead

Rules of Thumb	experience-based method
Maximize data locality	· don't fetch data you already have (no redundant data) · reconstruct computation to reuse data promptly ↳ 너무 fine-grained task 나누면 locality 떨어지니까 주의 ☞) communication 시간 줄도 있는! locality ↑ communication ↓
Minimize communication volume	· Partition interaction graph to minimize edge crossings ☆ · 그러려면 communication data size ⊕ frequency 둘다 control 필요
Minimize communication frequency	chunk 이용하기 ← size 좀 키워가지고 · aggregate message where possible - use batch (chunk) - saturate memory bandwidth 제일 Best
Minimize contention/hot-spot Master + slave 에서도 발생 가능	· no contention on use of same network link / memory block - can occur bottleneck · no specific process heavily-loaded (hot-spot) - can be load-imbalancing and idling · decentralized techniques - centralized can make bottleneck on master thread

Minimizing Interaction Overhead (Cont.)

Techniques	
Overlap Communication-Computation	Single-thread · use non-blocking communication primitives Thread switch X - 즉, I/O processing 동안 CPU computation 할 수 있도록 interrupt 같은 policy 적용! - overlap communication with your own computation .. 그래야 context switch ↓ - one-sided communication (R/W) : prefetch remote data to hide latency multi-thread · multithread code : overlap communication with another thread's computation
replicate data/computation to reduce communication	: communication을 줄이기 위해서 애초에 모든 process가 copy를 갖고 있는 것 - 예) dense matrix multiplication에서 vector b copy 해두는 것! Read-only는 괜찮지만 write하면 이제 inconsistency 발생
Use collective interaction instead peer to peer	: data transfer나 contention overhead를 줄일 수 있다
Issue Multiple communications	: reduces exposed latency. 즉, send를 한꺼번에 함으로써 pipelining 할 수 있는 건가? 일단 동시에 보내니까 serial 아니니까!

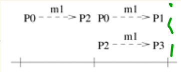
Overlapping Communications

- P0 broadcast data to others (P1, P2, P3)

  4 message

Step-wise algorithm

- commonly used algorithm



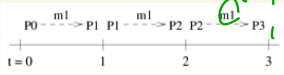
- takes 8 time to broadcasting end

m2 전달 끝

m1 전달 끝

naive algorithm

- 꼬리물기 !



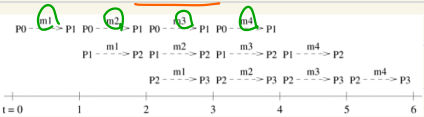
- takes 12 time for broadcasting

pipelined naive algorithm

뭔가가지고 있게 된다면, 바로작동가능!?

- 꼬리물기는 맞는데, pipelining하는 것!

pipelined fashion



- takes 6 time for broadcasting