Large-Scale Graph Analysis

패트

Large Graph

우리가 PS에서 흔히 다루는 그래프의 크기: |E| < 100만

그런데...

- 페이스북 월간 유저 수는 약 30억 명
- 구글이 인덱싱한 웹 페이지의 갯수는 수천억 개
- |E| = |V| * 수십~수백

온갖 거대한 그래프를 분석해야만 한다!

ex) Google의 PageRank, 페이스북의 “친구의 친구”, 기타 등등...

PS에서 하던대로 하면 어떻게 될까?

|V| = 1000만, |E| = 10억이라 하자

인접 리스트에 저장하면 **최소** 10억 * 4바이트 = 4GB

실제로는 수십~수백GB 이상의 규모 → **Out of memory!**

무지성 해결책: Swap space 이용

장점

- 일단 안 죽고 돌아가긴 한다
- 기존 코드를 전혀 수정하지 않아도 된다

문제점

그래프의 낮은 locality에 의한 **매우 심각**한 성능 저하 발생 (thrashing)

시뮬레이션: BFS

1번 정점부터 탐색

인접 정점 목록: 100만번, 200만번, 300만번 -> 큐에 넣음

다음 순서인 100만번 정점

100만번 정점의 인접 정점 목록이 메모리에 없으니 디스크에서 불러옴

인접 정점 목록 큐에 넣음

다음 순서인 200만번 정점의 인접 정점 목록도 디스크에서 불러와야 함

...

무수한 디스크에서의 로딩 발생

그게 그렇게 큰 문제인가?

메모리 읽기 레이턴시 = 100 ns

SSD 읽기 레이턴시 = 150,000 ns

디스크 읽기 레이턴시 = 10,000,000 ns 이상

최소 **수천배** 느려진다.

업계의 해결책

분산 컴퓨팅

“우리 컴퓨터 많잖아?”

해결해야 할 문제점

- 온갖 알고리즘을 돌려야 한다
- 그런데 분산 컴퓨팅은 신경쓸 게 너무 많다 (태스크 분배, 장애 대응 등)
- 알고리즘 하나 짤 때마다 그걸 일일히 신경쓰긴 힘들어...

MapReduce [1, 2]

OSDI ’04

빅데이터의 분산 처리를 위해 구글에서 개발

Lisp의 map과 reduce에서 착안

map과 reduce에 사용할 함수만 작성하면 병렬 처리, 장애 대응, I/O 스케쥴링, 모니터링을 MapReduce 프레임워크가 다 해준다!

MapReduce 개념 이해하기

문제 상황: 주어진 배열의 모든 원소의 제곱의 합을 구하여라.

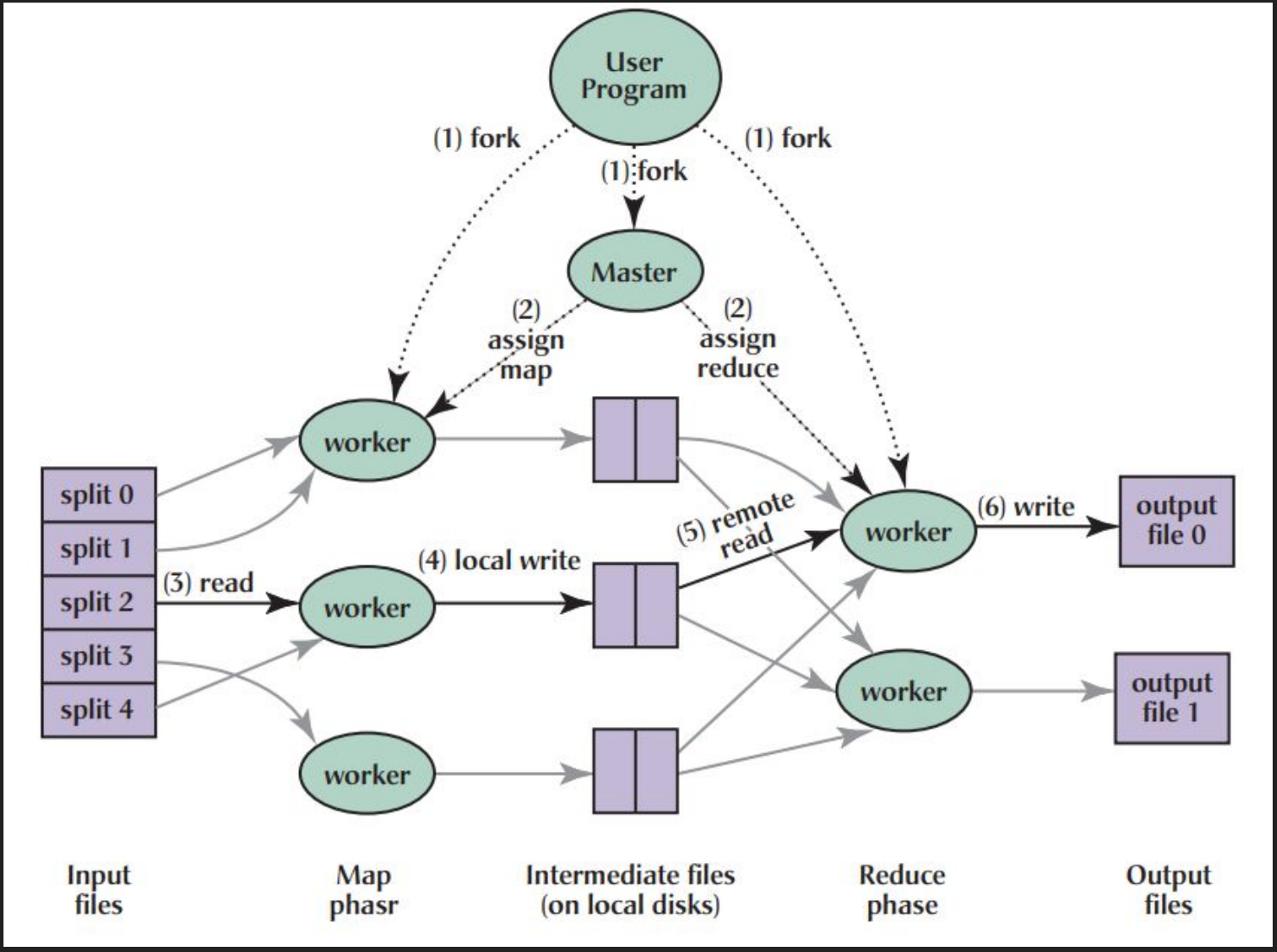
예제: [1, 2, 3] -> 1*1 + 2*2 + 3*3 = 14

map 함수: f(x) = x*x

reduce 함수: g(x, y) = x + y

[1, 2, 3] → [f(1), f(2), f(3)] → g(g(f(1), f(2)), f(3))

MapReduce와 분산 컴퓨팅



MapReduce의 문제점

빅데이터 처리에 좋은 건 맞는데, 그래프 처리에 특화되어 있지는 않다

그래프를 **어떻게** 쪼갤 것인가? → 매우 어려운 문제

그래프 알고리즘은 수많은 join이 발생하는 것도 문제

Pregel [3]

SIGMOD ’10

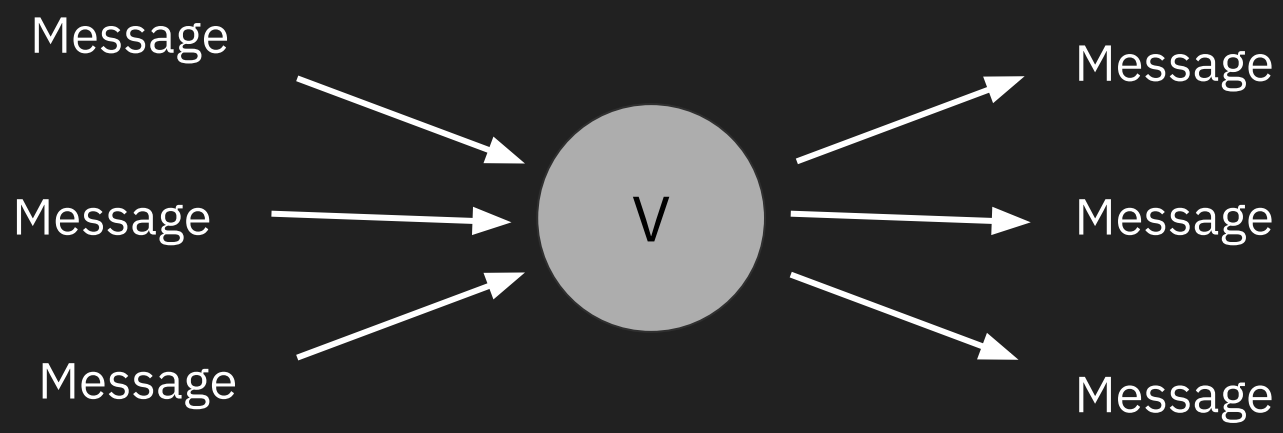
이번에도 구글!

대규모 분산 컴퓨팅 환경, 거대한 그래프에 특화

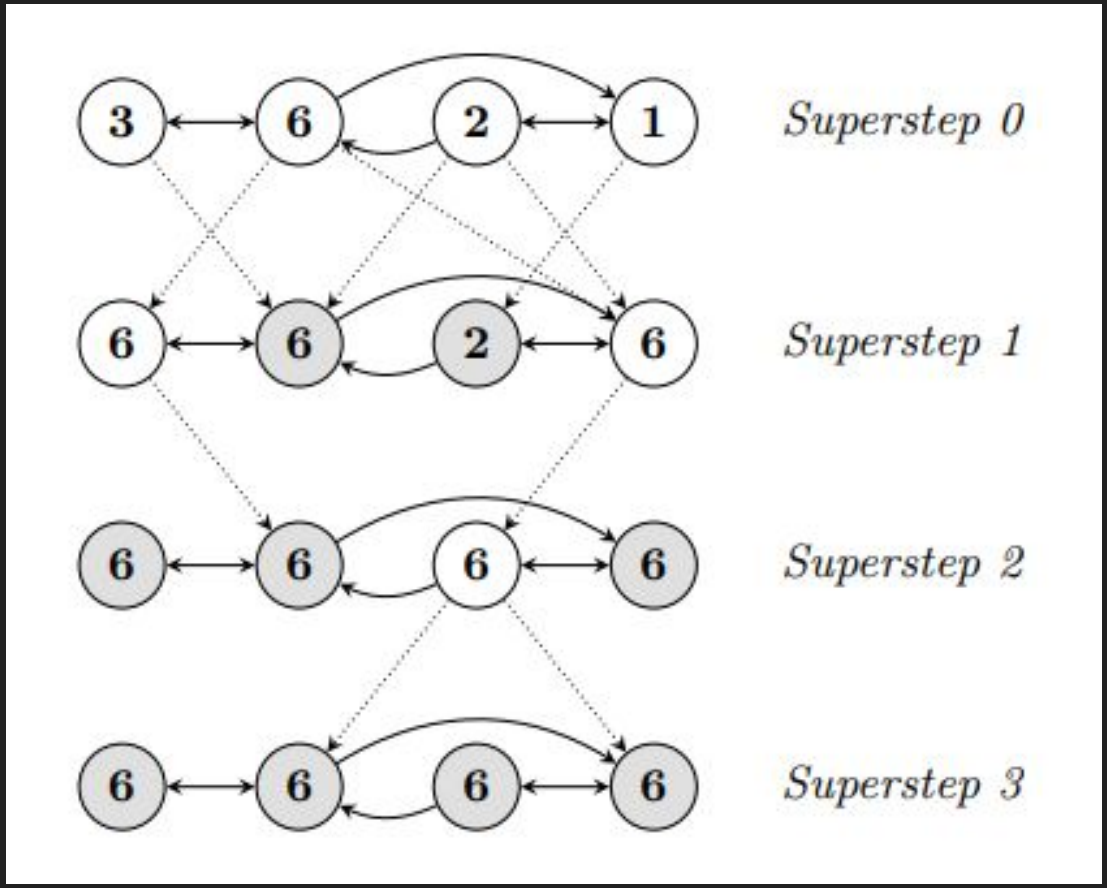
Pregel 모델: vertex와 superstep

Vertex 단위의 그래프 분할 (vertex-centric)

Superstep 단위의 알고리즘 실행



Pregel 예시: 최대값 찾기



그리고...

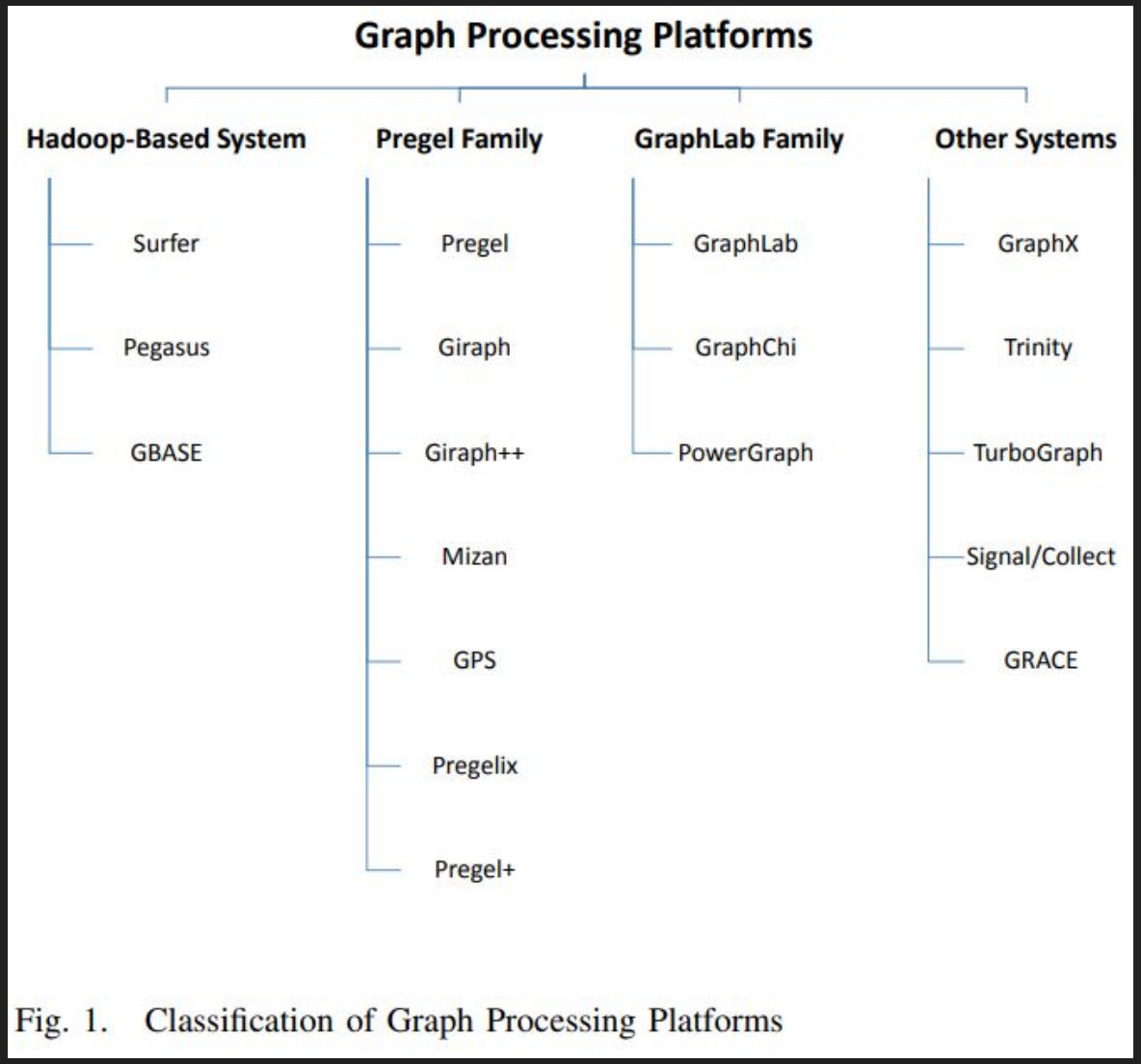


Fig. 1. Classification of Graph Processing Platforms

Batarfi, Omar, et al. "Large scale graph processing systems: survey and an experimental evaluation." *Cluster Computing* 18.3 (2015): 1189-1213.

GraphChi [4]

OSDI ’12

학계의 새로운 시도: 컴퓨터 한 대로는 안될까?

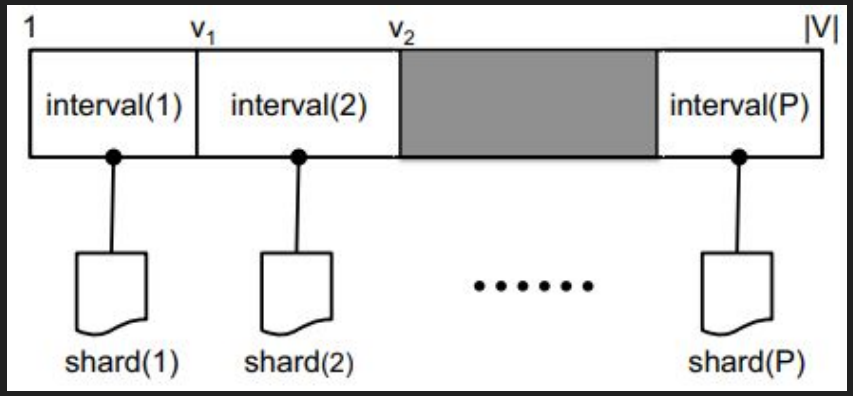
디스크를 이용한 무지성 메모리 확장은 안된다는 것은 이미 알고 있다...

I/O 횟수를 최소화하는 좀 더 똑똑한 방법이 없을까?

GraphChi: 기본 가정

1. 메모리가 작기 때문에 당연히 그래프 전체를 올릴 수 없다.
2. 심지어 모든 정점의 값을 메모리에 올릴 수도 없다.
3. **단일 정점**에 대한 모든 정보는 메모리에 올라간다.

GraphChi: Shard

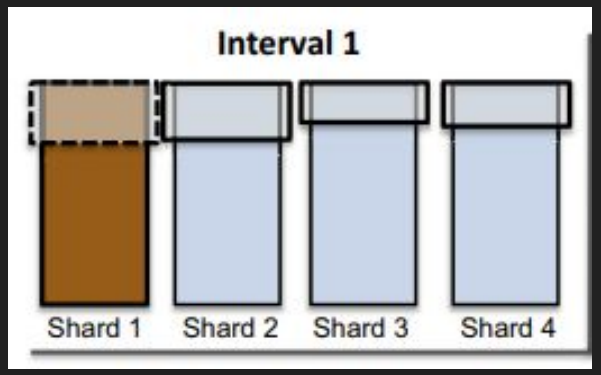


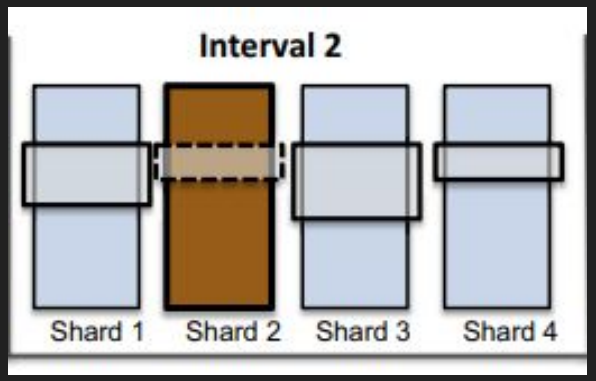
각 shard는 해당하는 interval으로 들어오는 모든 in-edge를 저장

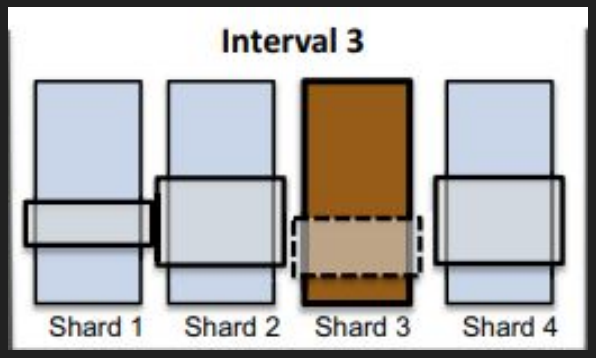
interval을 나누는 기준은 하나의 shard가 전부 메모리에 올라가도록 함

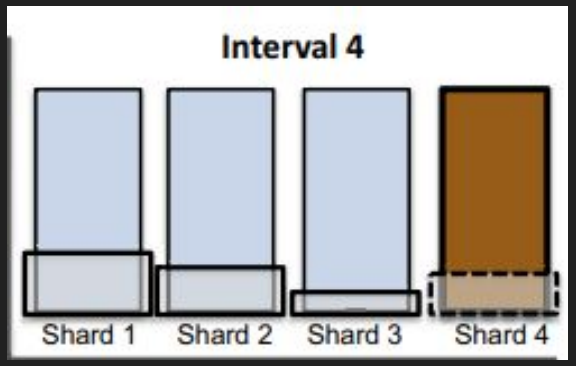
※ 전처리할 때 shard는 시작 vertex 기준으로 미리 정렬해둠

GraphChi: Parallel Sliding Window

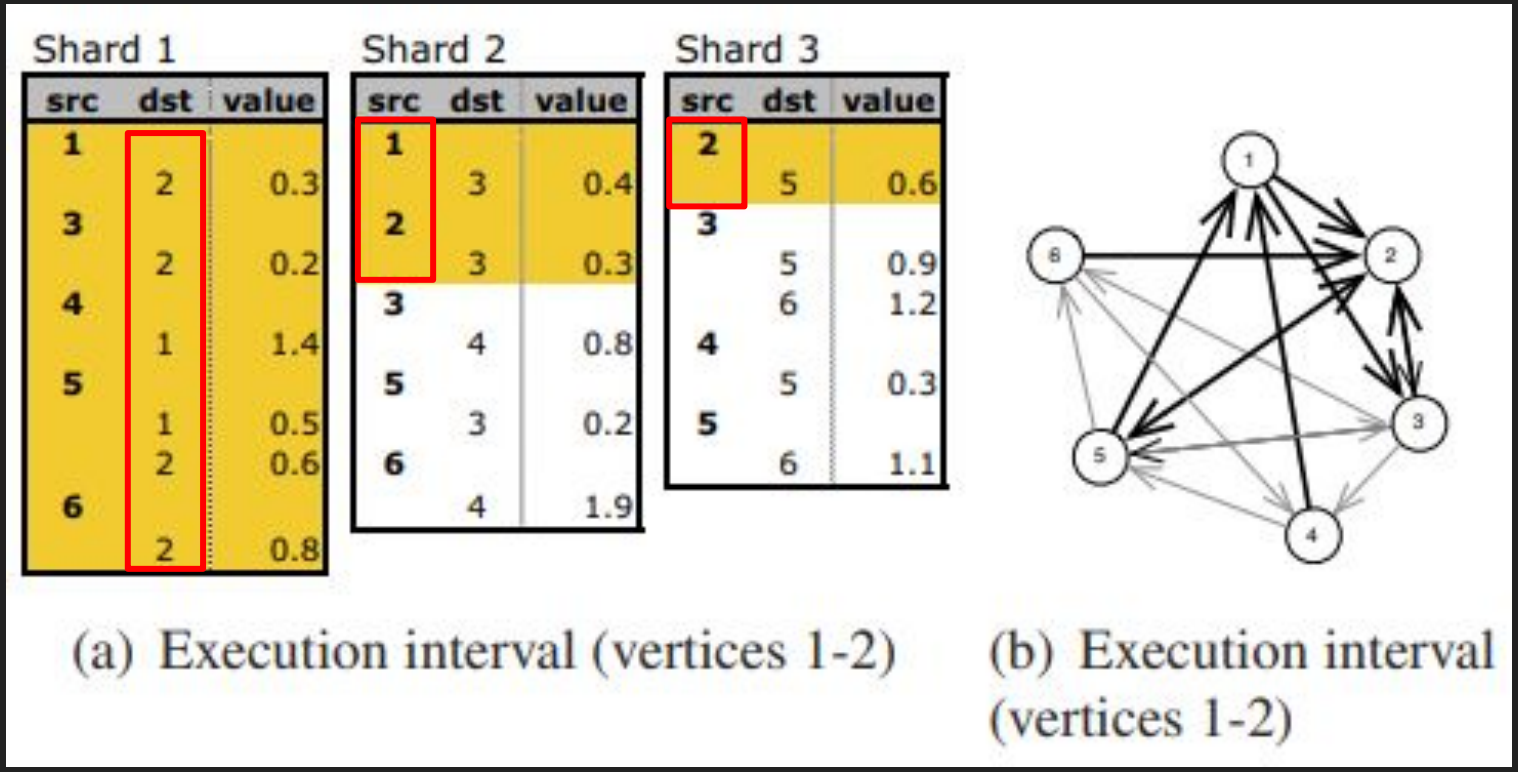








GraphChi: Parallel Sliding Window



Shard 1

src	dst	value
1	2	0.3
3	2	0.2
4	1	1.4
5	1	0.5
	2	0.6
6	2	0.8

Shard 2

src	dst	value
1	3	0.4
2	3	0.3
3	4	0.8
5	3	0.2
6	4	1.9

Shard 3

src	dst	value
2	5	0.6
3	5	0.9
	6	1.2
4	5	0.3
5	6	1.1

(a) Execution interval (vertices 1-2)

(b) Execution interval (vertices 1-2)

GraphChi: 결론

Interval 개수를 P개라 하면, 총 디스크 I/O 발생 횟수는 P^2번에 불과

심지어 전부 sequential → HDD와 SSD 모두 문제 없다

→ 단일 PC임에도 분산 컴퓨팅 환경과 비교해 꿀리지 않는 성능

궁금한 점: 이걸로 알고 구현이 다 될까?

Evaluation: Is PSW expressive enough?

Graph Mining
- Connected components
- Approx. shortest paths
- Triangle counting
- Community Detection

SpMV
- PageRank
- Generic

Recommendations
- Random walks

Collaborative Filtering
(by Danny Bickson)
- ALS
- SGD
- Sparse-ALS
- SVD, SVD++
- Item-CF

Probabilistic Graphical Models
- Belief Propagation

Algorithms implemented for GraphChi (Oct 2012)

Carnegie Mellon

X-Stream [5]

SOSP ’13

GraphChi와 마찬가지로 단일 PC에서 동작

Edge-centric 이라는 새로운 방법 제시

... 했으나 Vertex-centric가 결국 대세가 됨

FlashGraph [6]

FAST ’15

이젠 SSD가 대세니 이를 기반으로 최적화해보자!

- 특수한 파일 시스템 개발 (SAFS)
- 그 위에서 동작하는 그래프 엔진 개발 (FlashGraph)
- 그래프 연산과 I/O를 오버래핑
- 필요한 edge만 SSD에서 가져옴
- 기타 등등...

그리고 무수히 많은 논문들...

https://people.csail.mit.edu/jshun/graph.shtml

그래프 프레임워크 관련 논문만 100개 이상

다양한 세부 토픽

- 메모리 그렇게 비싸지 않은데 수백 GB 정도는 그냥 달아도 되잖아?
- GPU 써보는 건 어때?
- 그래프가 변하는 상황도 있잖아?
- 알고리즘 하나 잡아서 더 최적화할 수도 있지 않을까?
- 하드웨어 가속기를 붙일 수 있을까?
- 기타 등등...

마무리하며...

전하고 싶었던 메시지

- 컴퓨터 구조와 OS에 관한 이해는 최적화에 도움이 된다
- 좋은 어플리케이션을 만들기는 어렵다
- 좋은 프레임워크를 만들기는 훨씬 어렵다

- 학계의 발전: 새로운 아이디어와 점진적 발전
- 대학원 쉽지 않다

Reference

[1] MapReduce: Simplified Data Processing on Large Clusters

[2] MapReduce for Graph Algorithms Modeling & Approach

[3] Pregel: A System for Large-Scale Graph Processing

[4] GraphChi: Large-Scale Graph Computation on Just a PC

[5] X-Stream: Edge-centric Graph Processing using Streaming Partitions

[6] FlashGraph: Processing Billion-Node Graphs on an Array of Commodity SSDs