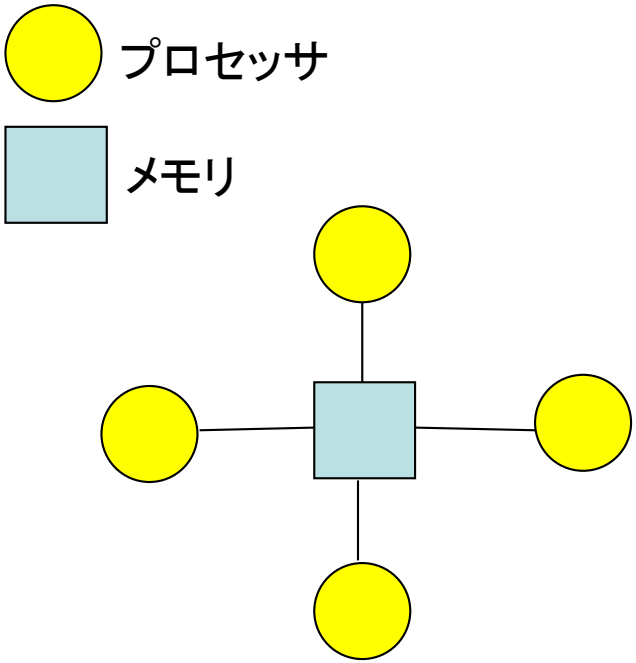
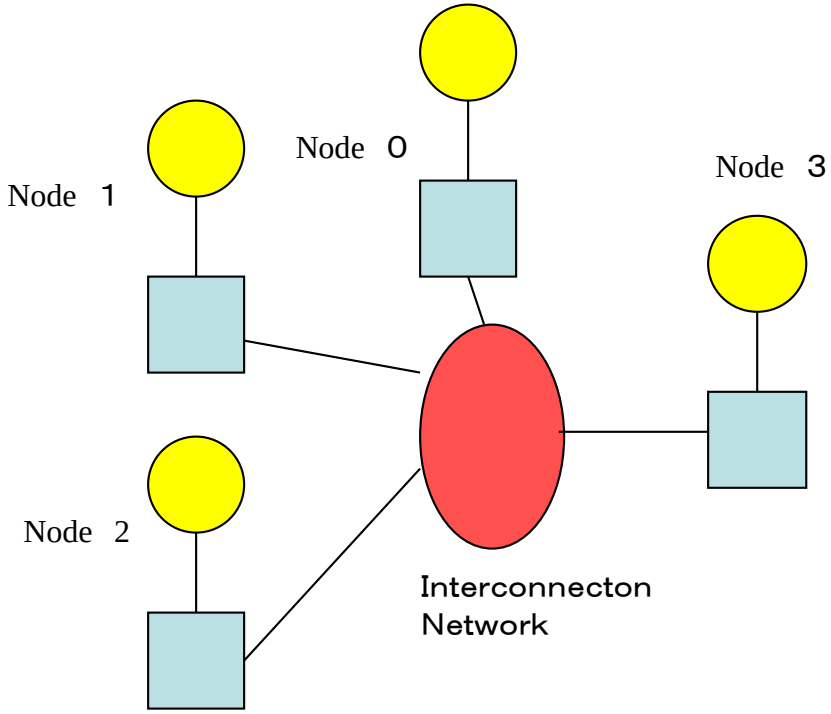
分散共有メモリ型計算機とクラスタ

慶應義塾大学理工学部
天野英晴
hunga@am. ics. keio. ac. jp

集中メモリ型と分散メモリ型



メモリが一か所に集中
UMA(Uniform memory access model)
いわゆるマルチコア



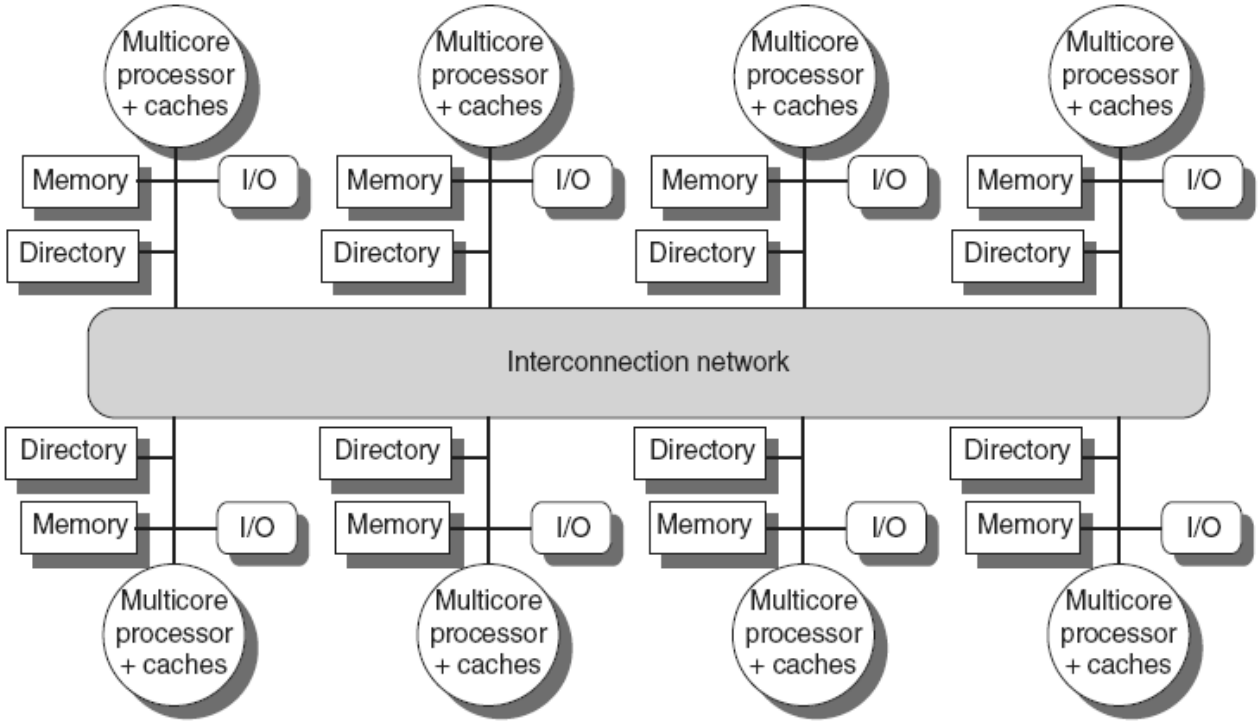
メモリが分散
NUMA(Non-Uniform memory access model)

NUMAの分類

- 単純なNUMA
  - 単一なアドレスでメモリをアクセスできるが、キャッシュはできない
  - できても一貫性は保持してくれない
- CC-NUMA (Cache Coherent) NUMA
  - キャッシュの一貫性を保持してくれる
- 大規模なNUMAは、一部CC-NUMAになっている

典型的な分散メモリ型

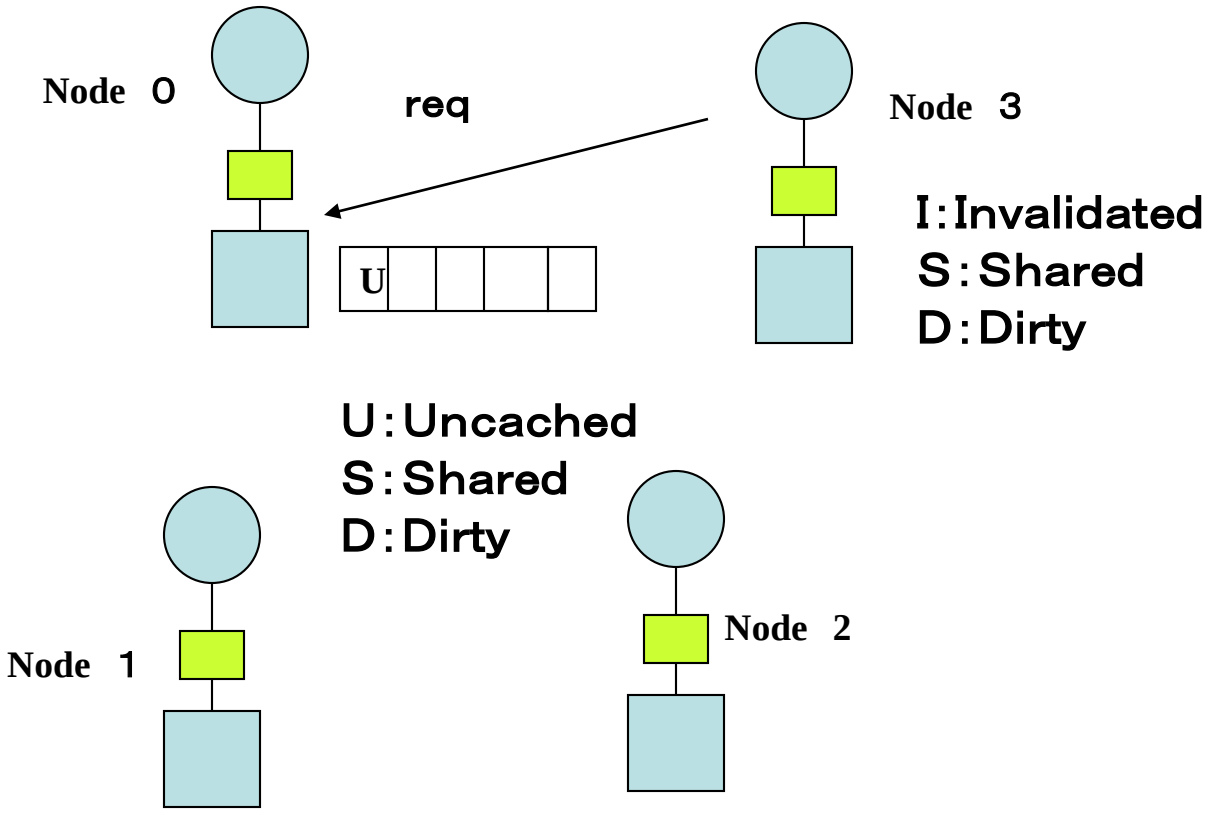
IBM Power 7
AMD Opteron 8430



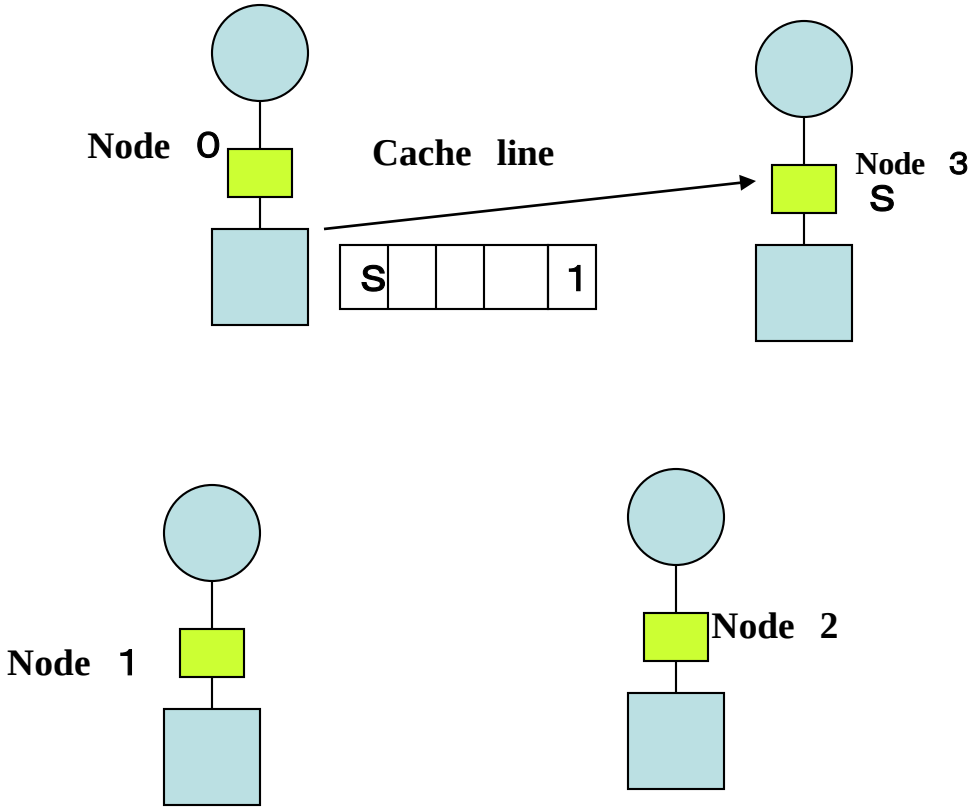
ディレクトリ方式のキャッシュ

- ホームメモリが共有情報をディレクトリで管理
- ノード間のメッセージ交換でキャッシュの一致を制御
- プロトコル自体はスヌープ方式と似ている
- バスがないので常にディレクトリをメッセージでアクセスする

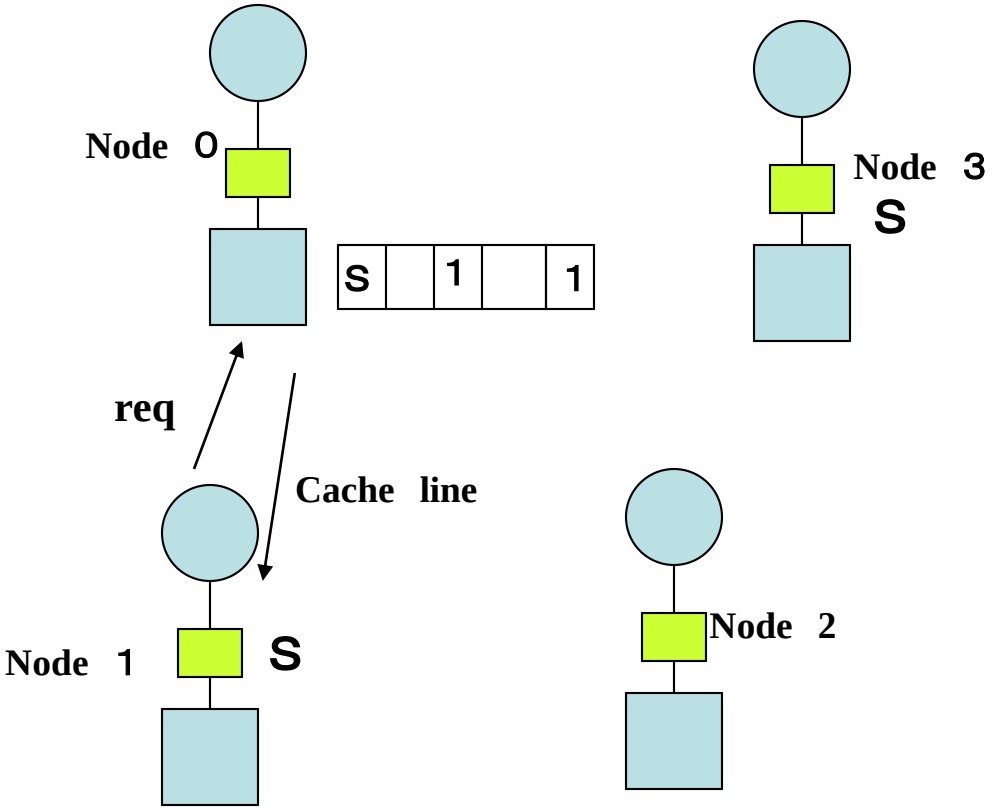
キャッシュの制御（Node 3読み出し）



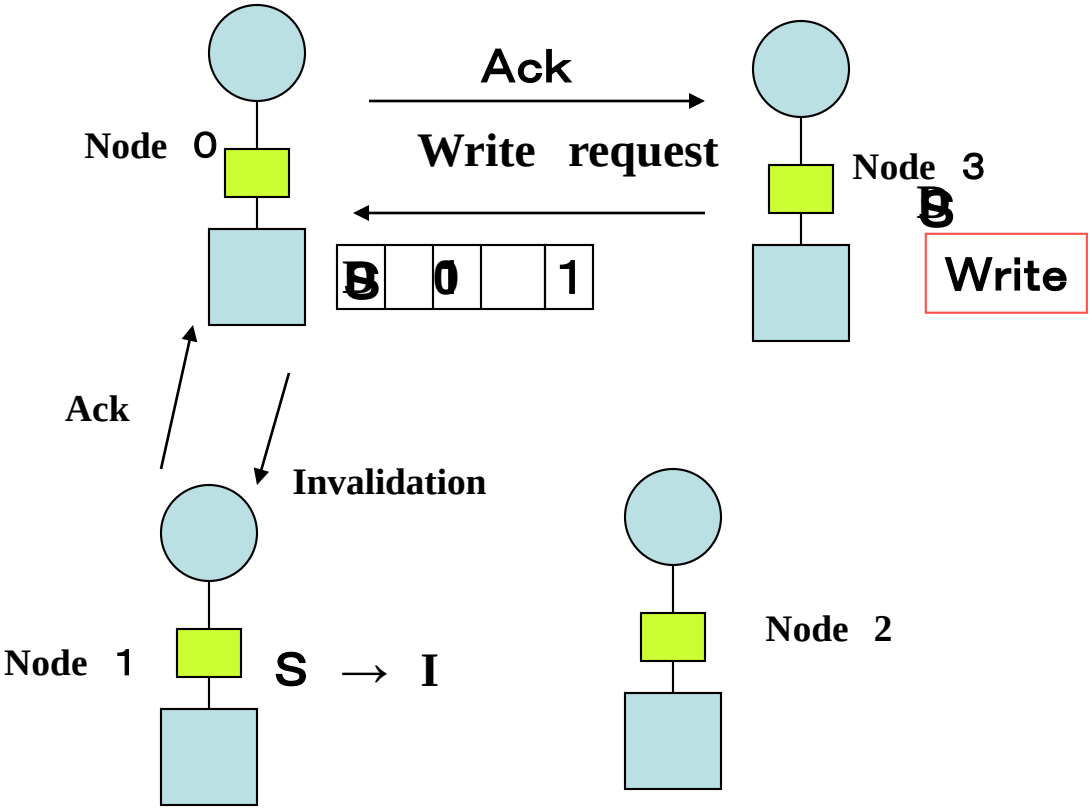
キャッシュの制御（Node 3読み出し）



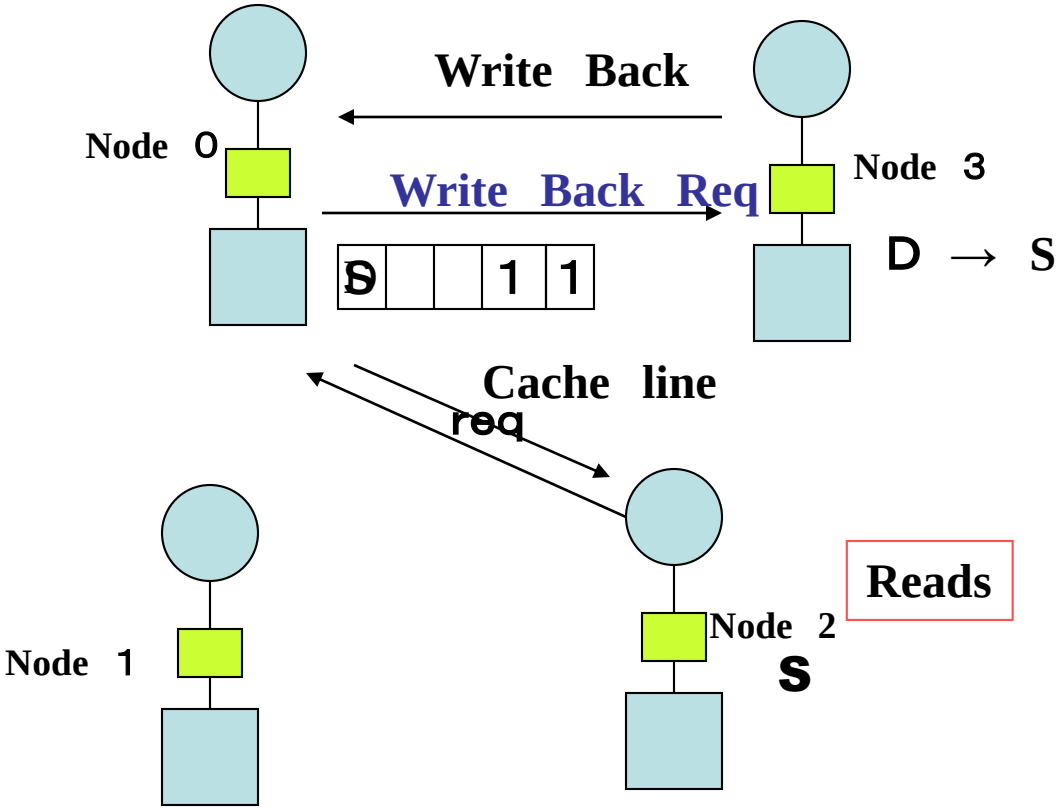
キャッシュの制御（Node 1読み出し）



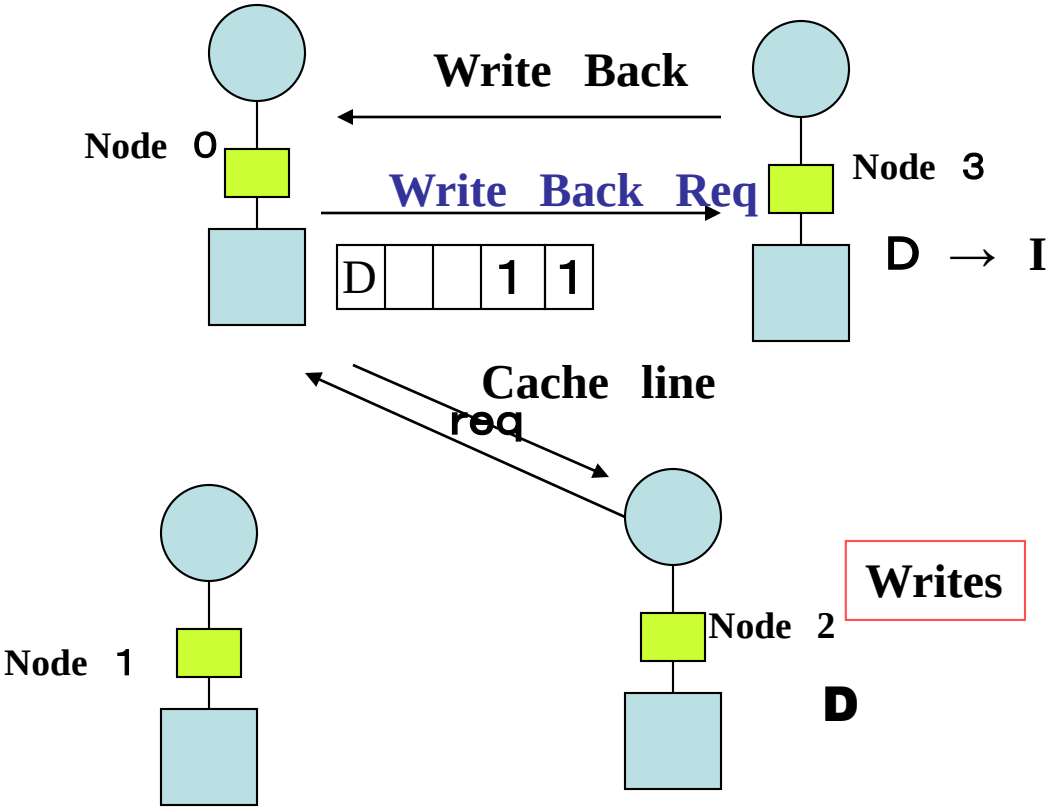
キャッシュの制御（Node 3書込み）



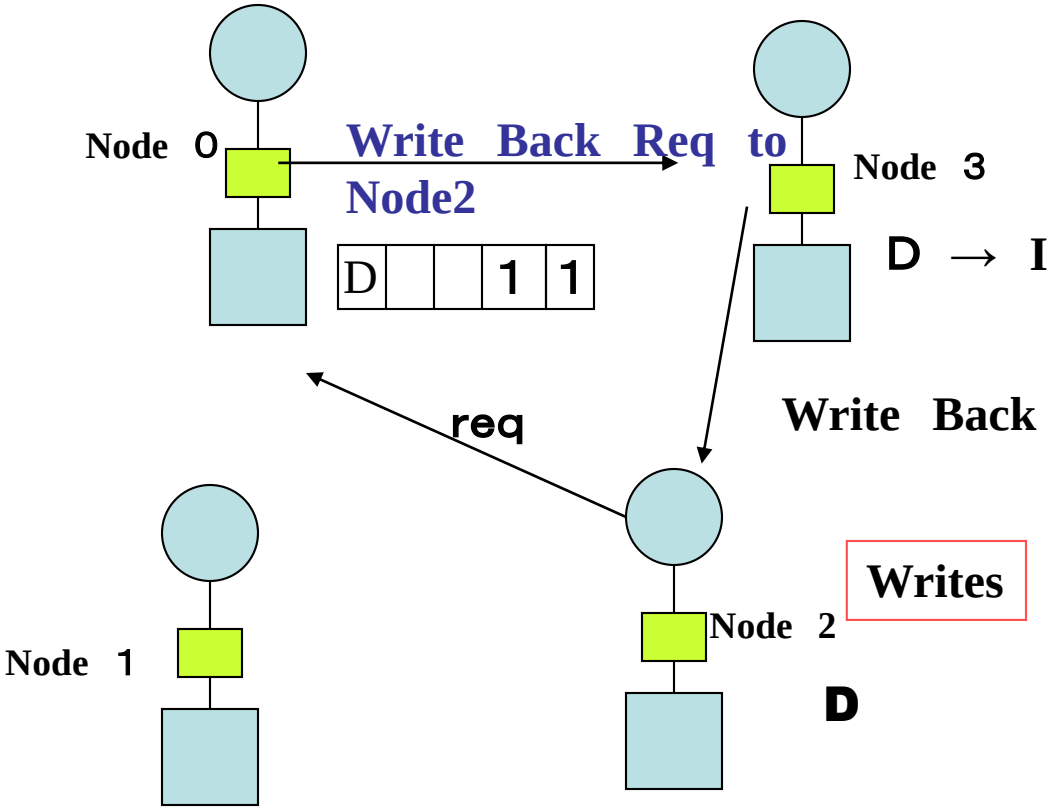
キャッシュの制御（Node 2読み出し）



キャッシュの制御（Node 2書込み）

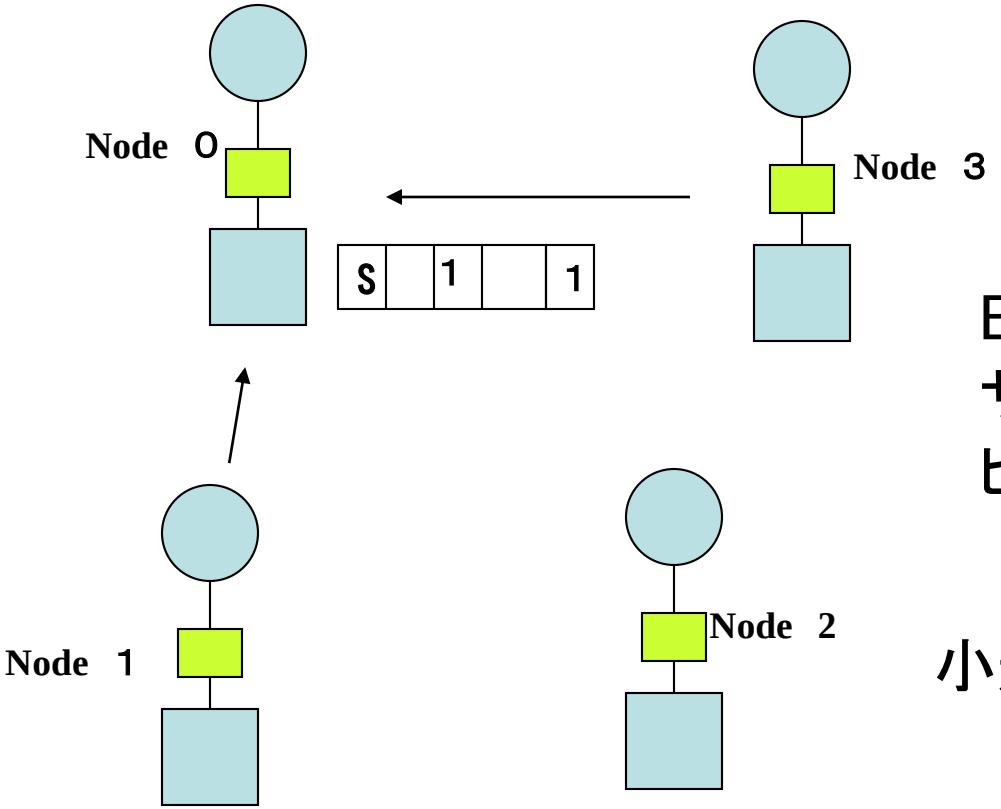


三角データ交換



Dのキャッシュが要求元に直接データを送る

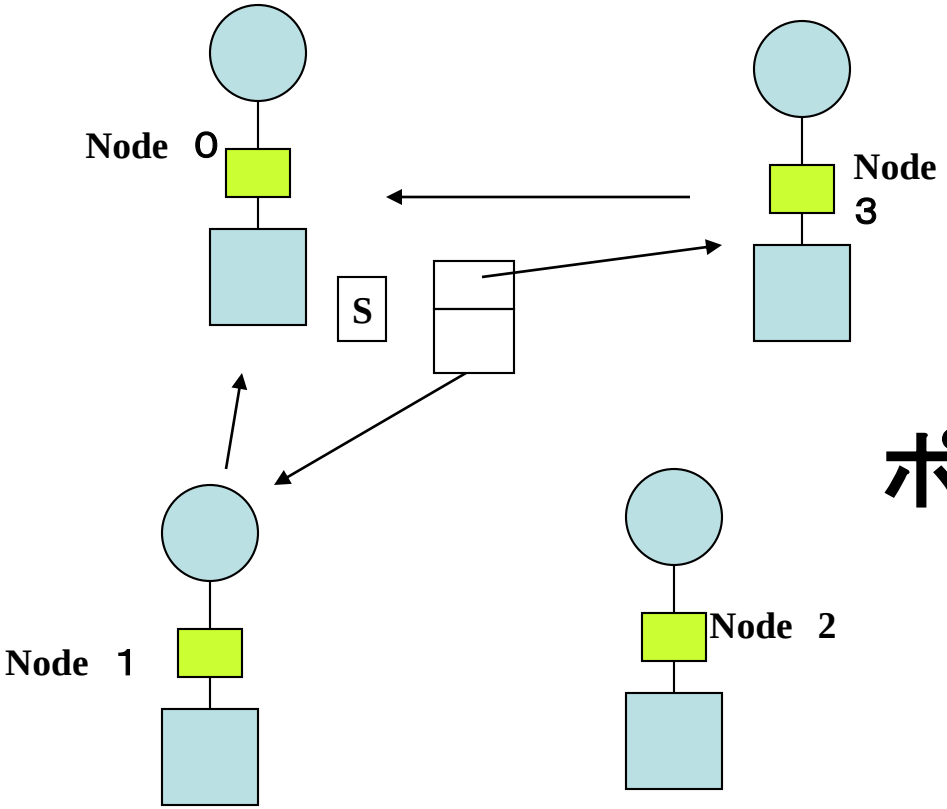
フルマップ方式



Bit = Nodes
サイズが大きいと
ビット数が増大

小規模向き

リミテッドポインタ

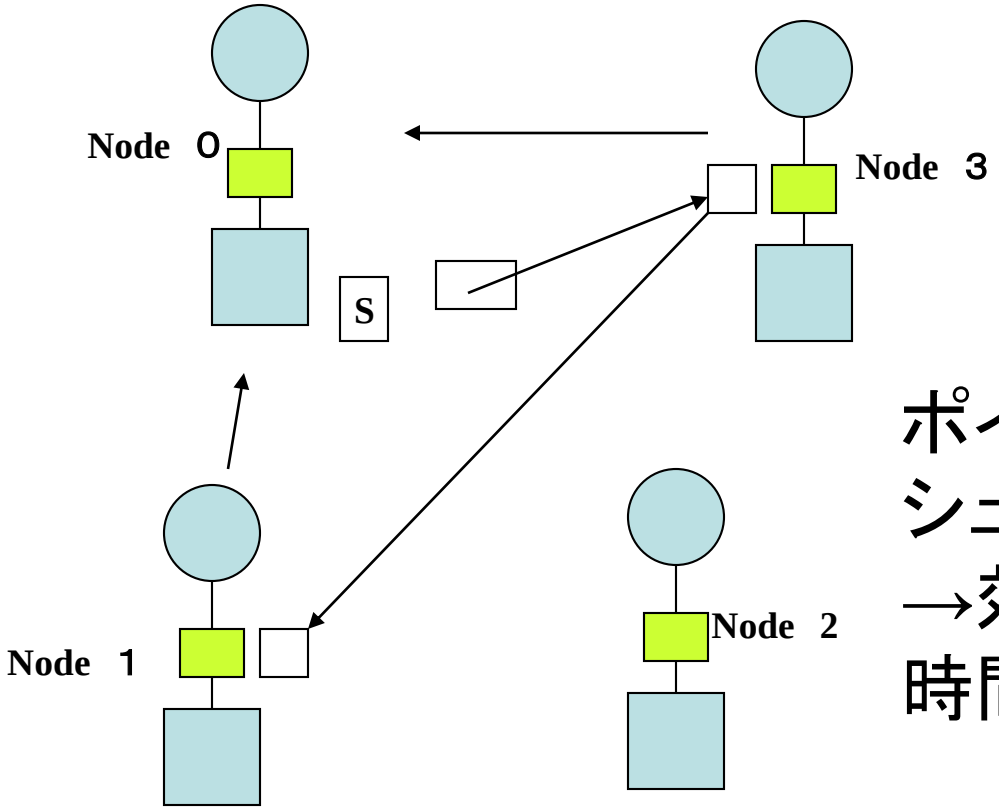


ポインタの利用

リミテッドポインタ

- 少数のポインタで共有関係を保持する
  - 実際に多数のノードで共有されるブロックはさほど多くない
- 溢れたらどうするか？
  - 無効化する（追い出す）
  - メッセージをブロードキャストに切り替える
    - 不要なメッセージは単に捨てれば良い
  - ソフトウェアを呼び出して何とかしてもらう

キャッシュ間のリンクを構築



ポインタをキャッシュ上の置ける
→効率が良い
時間が掛かる

ディレクトリをキャッシュに持たせる

- どこかのノードでキャッシュされているブロックは必ずホームメモリのキャッシュにも置く
  →ホームメモリの代わりをキャッシュがする

○高速なアクセスが可能

○全体のメモリ要求量が小さい

×キャッシュのコントローラが混雑

×キャッシュの利用効率が悪化

スヌープキャッシュとディレクトリキャッシュ

- スヌープキャッシュは共有バスのように皆が見る(スヌープ)ことのできる通信路が必要
  - 転送、キャッシュブロックの状態制御は分散的に行われる
  - 集中メモリシステムに向いている
- ディレクトリキャッシュは、ホームメモリのディレクトリが共有バスの代わりに管理センターの役割を果たす
  - メッセージを交換してブロックの状態制御を行う
  - 汎用性が高いが、メッセージ交換のコストは大きい

クラスタ

慶應義塾大学理工学部
天野英晴
hunga@am. ics. keio. ac. jp

NORA/NORMA

- 共有メモリを持たない
- 交信はメッセージのやりとりで行う
  - MPIが主に使われる
- 接続はGigabit EthernetやInfiniband
- 最近は多出力のスイッチを用いる
  - ハイラディックスネットワーク

データセンターなどで要求レベル並列性を処理

クラスタコンピューティング

Beowulf クラスタ

1994年NASA T.Sterling

- 安価で簡単に大規模並列計算環境を作ろう
  - コモディティのPCを利用
  - コモディティのネットワーク（Ethernet）を利用
  - コモディティのソフトウェア（Linux)を利用
  - PVMやMPIなどのメッセージパッシング型ライブラリでプログラム

→現在のClusterの元祖となった

現在のClusterは、InfinibandなどのSANを使うものも多いが、基本的に上記の原則を守っている

Infiniband

- System (Storage) Area Network (SAN)用.
- 8b/10b コードを利用.
- 様々な接続形態に対応.
- マルチキャスト可能.

	SDR	DDR	QDR
1X	2Gbit/s	4Gbit/s	8Gbit/s
4X	8Gbit/s	16Gbit/s	32Gbit/s
12X	24Gbit/s	48Gbit/s	96Gbit/s


RHiNET-2 cluster

WSC（Warehouse Scale Computing）

- 巨大なクラスタの集合体
  - Google, Amazon, Yahoo,.. etc.
  - 50000ノードを越える規模
  - 経済的かつ均質なノードから構成される
  - ソフトウェアで信頼性を保証
  - 電源供給、冷却システムが重要で高価
- クラウドコンピューティングを支える

WSCは単純なデータセンターと少し違う

- 均質な構造
  - 従来のデータセンターは、実は多様なクラスタからできている。ソフトウェアやアプリケーションパッケージも色々
  - WSCはホモジーニアスな専用のハードウェアを使う。ソフトウェアも自作かフリーソフトウェア
- コスト
  - データセンターは、管理用の人件費が大きい
  - WSCでは、管理用の人件費は切り詰められている。このためサーバーハードウェアが大きい

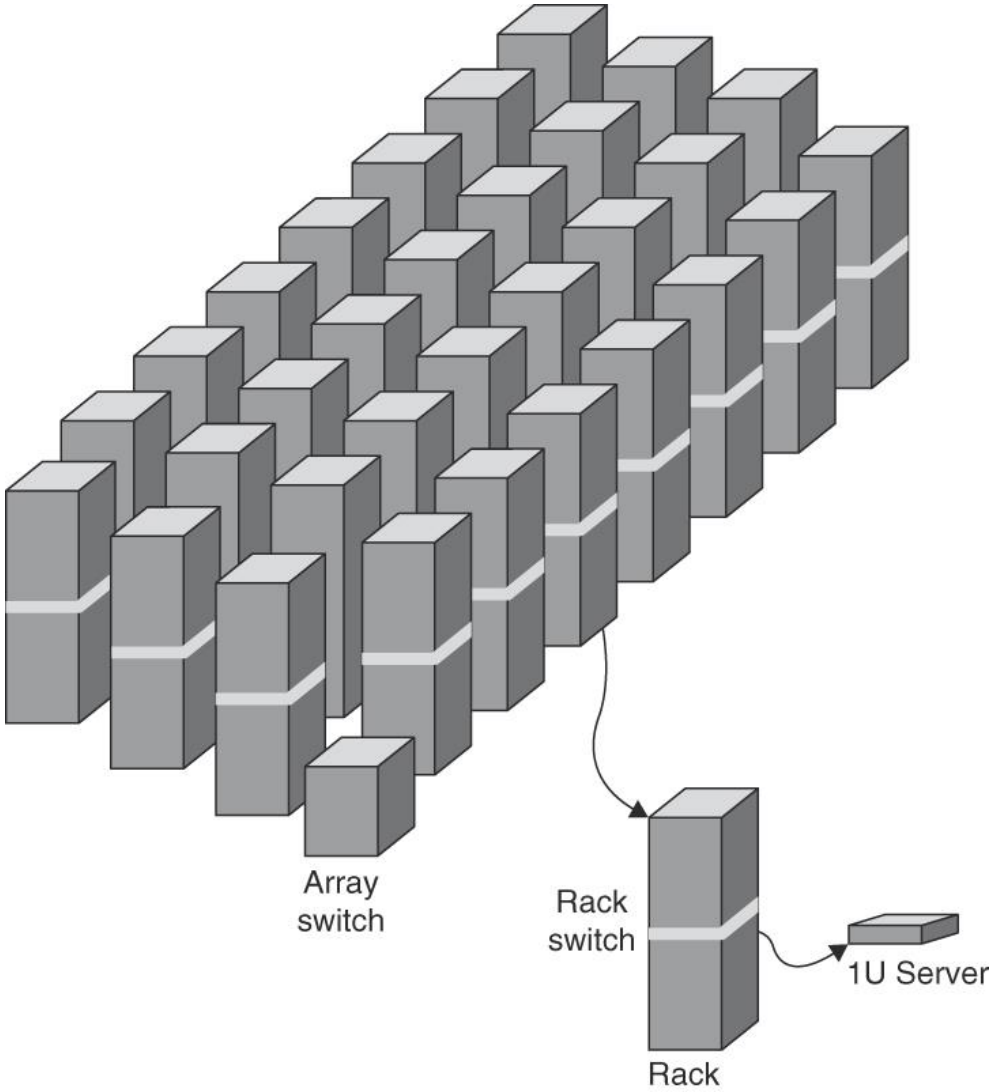


Figure 6.5 Hierarchy of switches in a WSC. (Based on Figure 1.2 of Barroso and Hölzle [2009].)

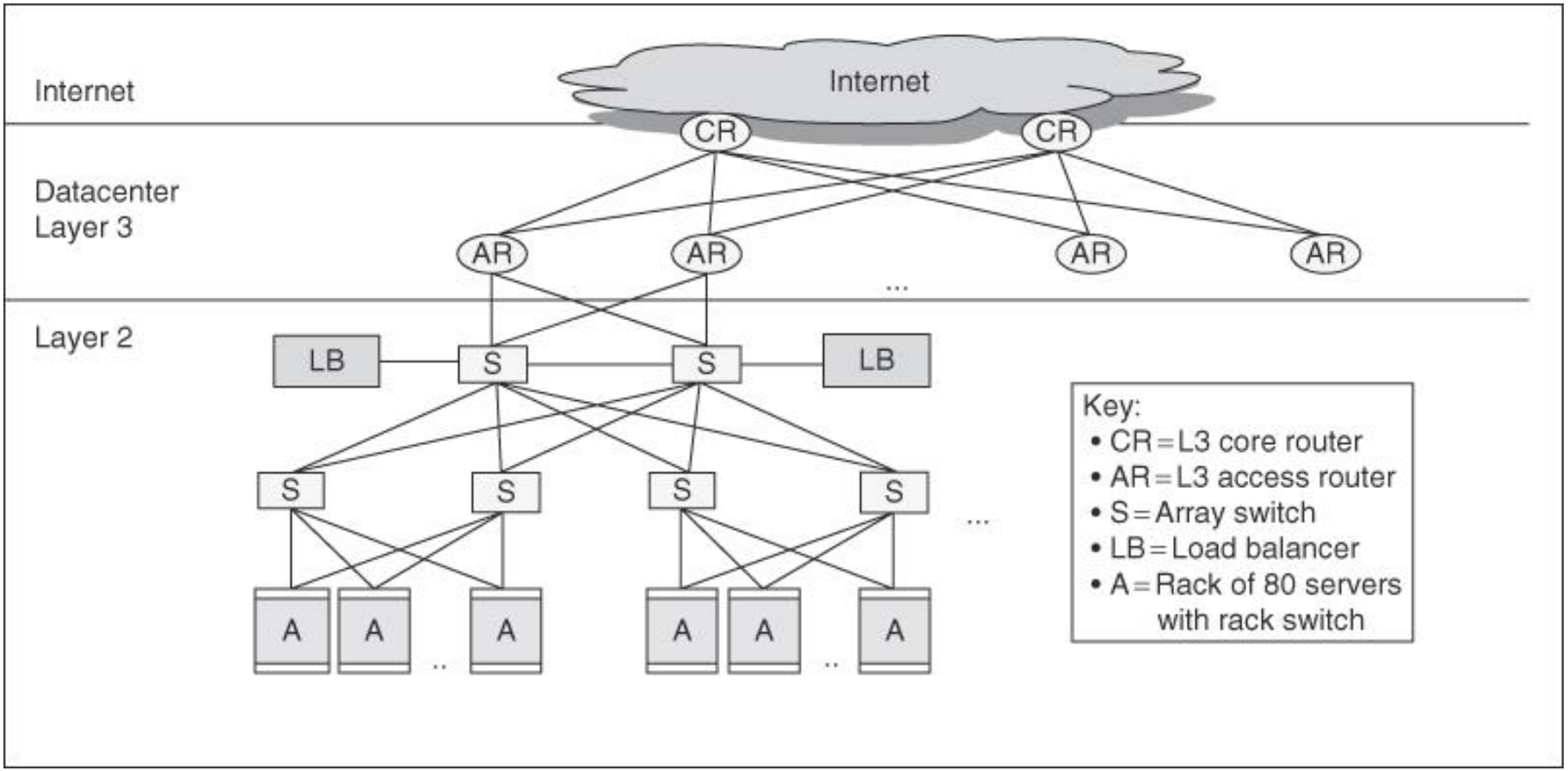


Figure 6.8 The Layer 3 network used to link arrays together and to the Internet [Greenberg et al. 2009]. Some WSCs use a separate *border router* to connect the Internet to the datacenter Layer 3 switches.

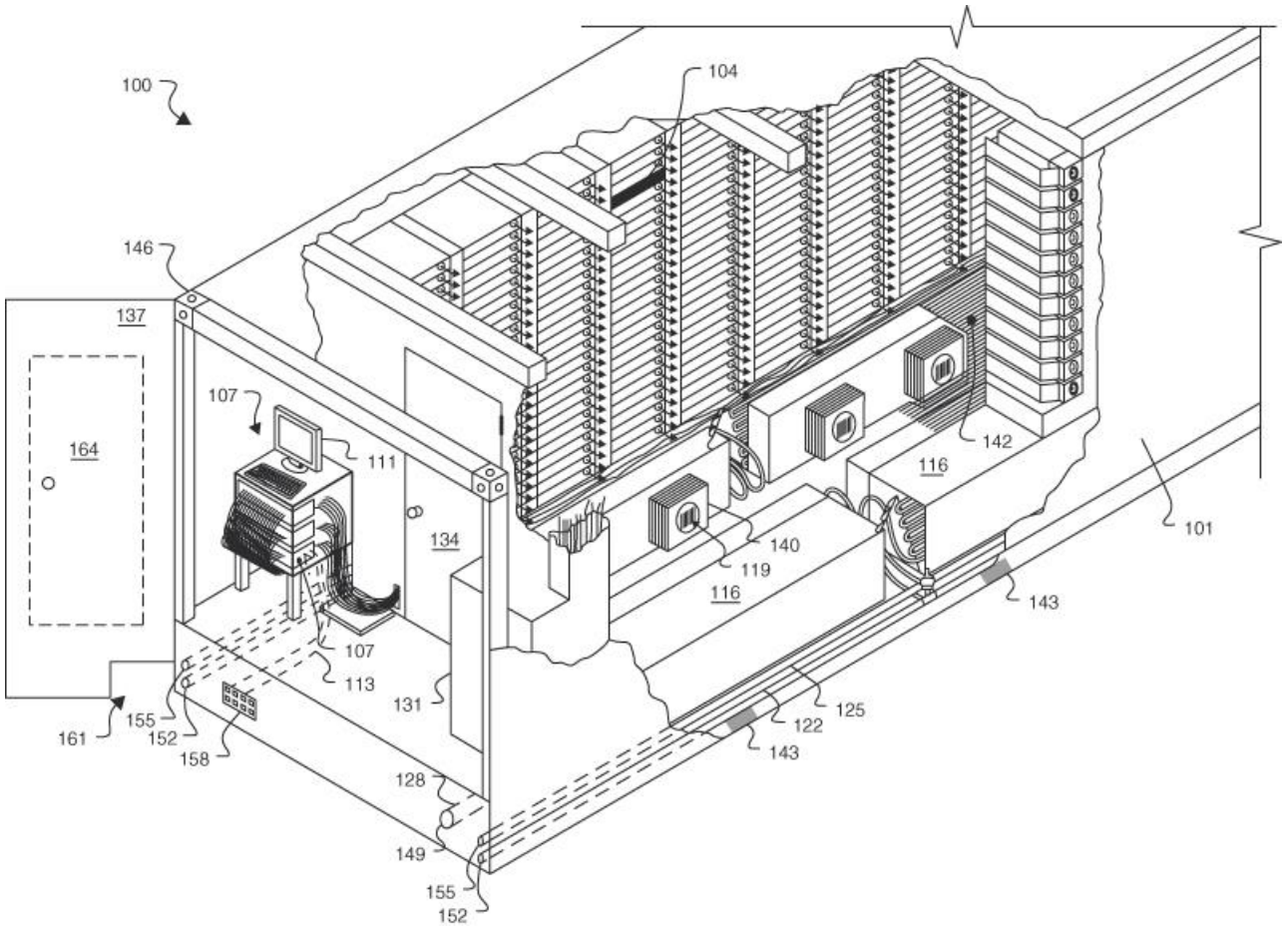


Figure 6.19 Google customizes a standard 1AAA container: 40 x 8 x 9.5 feet (12.2 x 2.4 x 2.9 meters). The servers are stacked up to 20 high in racks that form two long rows of 29 racks each, with one row on each side of the container. The cool aisle goes down the middle of the container, with the hot air return being on the outside. The hanging rack structure makes it easier to repair the cooling system without removing the servers. To allow people inside the container to repair components, it contains safety systems for fire detection and mist-based suppression, emergency egress and lighting, and emergency power shut-off. Containers also have many sensors: temperature, airflow pressure, air leak detection, and motion-sensing lighting. A video tour of the datacenter can be found at *http://www.google.com/corporate/green/datacenters/summit.html*. Microsoft, Yahoo!, and many others are now building modular datacenters based upon these ideas but they have stopped using ISO standard containers since the size is inconvenient.

演習

- CC-NUMAにおいて、home memoryのディレクトリ＋状態とキャッシュの状態を示せ
- ノード0のホームメモリに対して以下のアクセスが順にあったとする:
  - Node 1 reads
  - Node 2 reads
  - Node 1 writes
  - Node 2 writes