# 要約資料

- 本資料は下記論文を基にして作成されたものです。
  文書の内容の正確さは保証できないため、正確な知識を求める方は原文を参照してください

- WEB＋DB PLESS plusシリーズ
  [Web開発者のための]
  大規模サービス技術入門
- 2010年8月5日　初版　第1刷発行
- 著者：伊藤直也　田中慎司

# 大規模サービス技術について

理工学部　情報工学科
4年　080430054
鈴木一弘

# はじめに

- システムの巨大化への対応
  - Webサービスは成功にともない、成長し続ける
- 応急処置ではすぐに限界が訪れる
- サービス設計で気をつけるべきことを身につける
  - データが大きいときの基本的な考え方
- 題材…㈱はてなのWebサービス開発

# 大規模について

- サーバ台数…百～数千台
- データ規模…ギガバイト～テラバイト

- はてなのサービス(2009年夏頃)
  - 登録ユーザ100万人以上
  - 月間数十億アクセス
  - ピーク時の回線トラフィック量430Mbps
  - サーバ台数500台以上

- 超大規模サービス(参考)
  - GoogleやFacebookなどはサーバ台数数百万台、データ規模はテラバイト～ペタバイト

# メモリとディスク

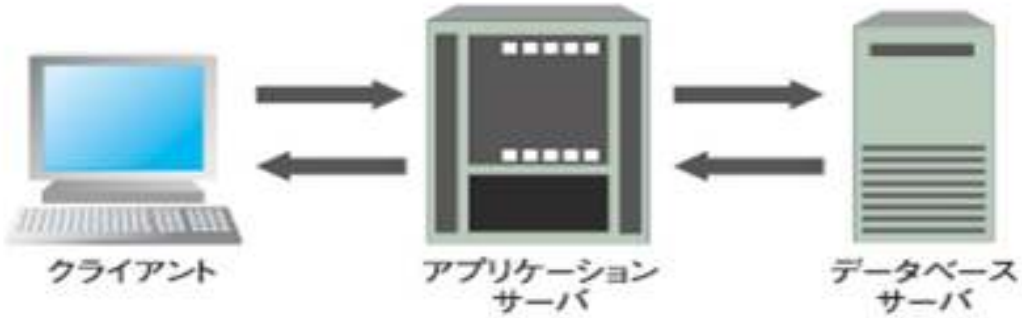
- 大規模データを扱う上で1番の難所はメモリ内で計算できないこと
- メモリ
  - 探索は電気的動作のみで高速(マイクロ秒単位)
  - 探索に使われるアルゴリズムやデータ構造がキャッシュに乗りやすいものだとさらに高速になる(ナノ秒単位)
  - 転送速度：7.5GB/秒
- ディスク
  - 探索は物理的動作が伴い、遅い(数ミリ秒)
  - 転送速度：58MB/秒
- メモリとディスクは探索速度で10万～100万倍以上、転送速度で100倍以上の差

# 速度差への対応

- Webサービスでは高価で速いハードウェアにする「スケールアップ」戦略よりも、どこかちょっと劣る点があっても旧版の安価なハードウェアを並べる「スケールアウト」戦略が有効
  - 値段と性能、速度は必ずしも比例しない
  - 負荷が少ない内は最小の投資で、必要に応じて拡張していく
  - ちょっとした用途でもコストがかからず簡単に用意することができる
- サーバを増やすことで、負荷を分散する

  →増やせば増やすだけ負荷は減らせる？

# CPU負荷とI/O負荷




- 負荷には大きく分けてCPU負荷とI/O負荷の2種類がある
- アプリケーションサーバにかかる負荷がCPU負荷
  →同じように仕事をするホストを増やすことで負荷分散が可能
- データベースサーバにかかる負荷がI/O負荷
  →DBサーバを増やした場合のデータの同期はどうするのか？
  →同期ができれば負荷は軽減？

# I/O対策

- ディスクを読み込むI/O負荷はできるだけ減らしたい
- OSのキャッシュはメモリを使うことでディスクアクセスを減らす
  - はてなのOSであるLinuxではメモリが空いていれば全部キャッシュする

→メモリを増やすことでキャッシュにできるだけ多くのデータを乗せ、I/O負荷を軽減できる

- 実際にはコストの面とのバランスをとってデータを圧縮することでキャッシュに乗せるのか、メモリの大きなサーバにするのかを決める

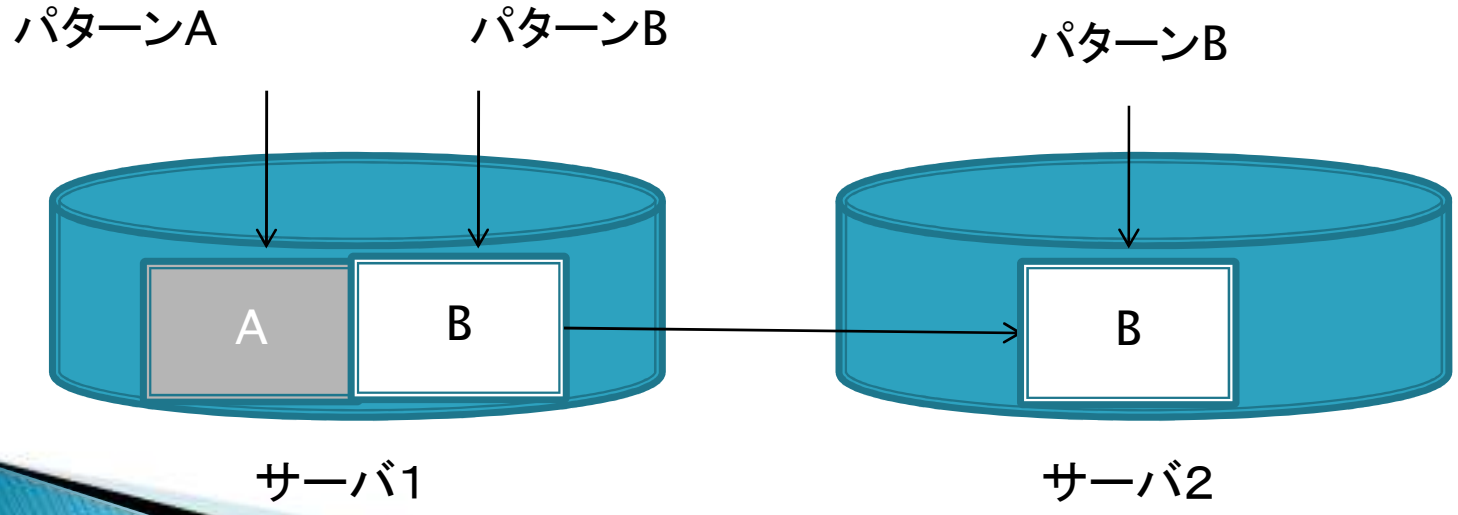
# DBサーバを増やす

- 大規模になることでキャッシュしきれない規模になってしまうことがある

→DBサーバを増やす必要が出てくる

→局所性を生かした分散

- アクセスパターンを考慮する



# 局所性を考慮した分散

- パーティショニング
  - データベースのテーブル単位での分割
  - 例：はてなブックマークにはentryテーブル、bookmarkテーブル、tagテーブル、keywordテーブルというテーブルがある
- データの途中での分割
  - 例えばIDがa～cで始まるひとのデータはサーバ1、d～hで始まる人はサーバ2…など
  - 実装は簡単
- リクエストパターンでの分割
  - 人間のユーザからならサーバ1、GoogleやYahooなどのbotからならサーバ2、一部機能のリクエストならサーバ3というふうに分割する

# キャッシュを考慮した運用

- OS起動後すぐにサーバを投入しない
  - キャッシュがたまっていないため、ディスクアクセスが発生し、はてなほど大規模になるとサーバが落ちてしまう
- 性能評価はキャッシュが最適化されてから性能評価を行う
  - キャッシュが乗っている時とそうじゃない時では性能や負荷は全く異なる

# 冗長化

- スケールアウトにともなって、台数が増えることで故障率があがる
- 分割にともなってサーバを増やす時、1台増やすだけではすまない
  - 各サーバは基本的に4台1セット
- マスタ/スレーブ
  - 1台が故障したら、サービス停止ではいけない
  - マスタに書き込まれた内容とおなじ内容をスレーブは自分自身に書き込む
  - サーバの参照はスレーブに行う

# 省力運用

- サーバ台数の増加に伴い、サーバの把握が困難になる
- 負荷、故障、ディスク容量、セキュリティ等の管理が必要

  →監視用ソフトウェアを使い自動化する

- しかし、セットアップや情報の確認は人間の役目であり、以下に人手をかけずに省力運用ができるかを追求していく必要がある

# 開発方法、開発者の統制

- 大規模サービスとなると1人での開発、運用が困難
  →複数の技術者で役割分担が必要

- 効率化を図るためのプログラミング言語や開発方法の標準化
- 各部門に特化した技術者の教育や、ルール遵守の管理などのマネージメント

# まとめ

- 大規模なサービスになるうえでの問題やそれに対応する為の技術について学んだ
  - CPU負荷とI/O負荷
  - 負荷分散
  - 冗長化
  - 省力運用
  - 開発方法の統制
- 今回の輪講では大規模サービスの開発を行う上での下地となる考え方や基本技術を学んだ。