計算モデル特論

国立情報学研究所

佐藤一郎

E-mail: ichiro@nii.ac.jp

自己紹介：佐藤一郎

- 国立情報学研究所・
  アーキテクチャ科学研究系・教授
- 国立大学法人総合研究大学院大学・
  複合科学研究科・情報学専攻・教授
- Rank Xerox Grenoble研究所客員研究員（1994-1

国立科学博物館（上野）

排出量取引の社会実験
（イトーヨーカドー）

実証実験（そごう横浜店）

宣伝

国立情報学研究所／国立大学法人 総合研究大学院大学
複合科学研究科 情報学専攻 学生募集

5年一貫制博士課程

大学の学部卒業と同等以上の学力を有する者を対象とし、標準5年間で博士の学位を取得

博士後期課程

大学院修士課程修了と同等以上の学力のある者を対象とし、標準3年間で博士の学位を取得

国立情報学研究所（NII）

講義の目標

- 計算モデルを通じて計算という概念を理解すること
- 計算の数学的な取り扱いに慣れること
- 計算モデルを通じて(計算)システムを捉えること
  - 複雑システム(並列・分散システム、相互作用)をモデル化
    - 例：クラウドコンピューティングは非同期計算モデルがベース
    - 例：Google MapReduceは関数型計算モデルがベース

- 評価：　　レポート（数回）（たぶん一回）

計算モデル

- 計算モデルの必要性
- 世の中の情報システム(コンピュータや生物)は非常に複雑
- この情報システムがどのような原理に基づくかを知るには数学的基盤を持つ抽象的な枠組みを導入する必要がある。
  - 情報システムの抽象的な枠組み＝計算モデル
- 計算モデルを通じて「計算」及び「情報システム」という概念を明確化

講義内容（今年）

- 計算モデルとは
- 並列分散計算のモデル
  - CCS (Calculus for Communicating Systems)
  - 操作意味論
  - プロセス等価性
  - Petri-net
  - 時相論理
- 非計算システムの計算モデル

講義内容（昔）

- 講義内容（予定）
- 計算モデル
  - 抽象機械モデル、関数型モデル、プロセス代数（プロセスカルキュラス）
- プログラミング言語意味論
  - 操作意味論、表示意味論、公理意味論
- 型理論
  - 型付きラムダ計算、多相型、オブジェクト指向型

代表的な計算モデル

- （逐次）計算モデル:
- 抽象機械計算モデル
- 関数型計算モデル
- 論理型計算モデル
- 項書き換え型計算モデル

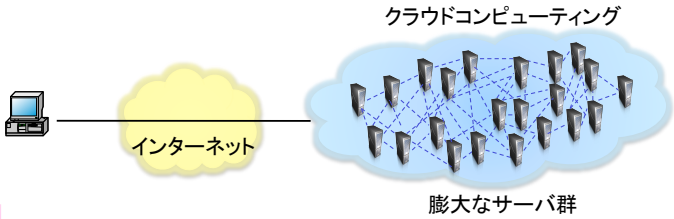
- （並列分散）計算モデル
- 抽象機械計算モデル
- プロセス代数（プロセスカルキュラス）
- イベント機械モデル（ペトリネット）

なぜ、並列分散計算モデルなのか

- ここ数年は並列分散計算モデルに特化しています。
  - 並列分散計算モデルとは
    - 複数同時に実行されるシステム
    - システム間通信
- 理由:
  - Webをはじめ通信する計算システムばかり
  - クラウドコンピューティングなど複数サーバによるシステム構成
  - PCやスマートフォンではマルチコアがあたりまえ
  - 実世界は複数の実体が同時並列に動いている

計算システムの変遷

- 逐次型計算システム(1950年)
- マルチタスク計算システム(1970年)
- クライアントサーバ計算システム(1990年)
- Webベース計算システム(2000年)
- クラウドコンピューティング(2008年)

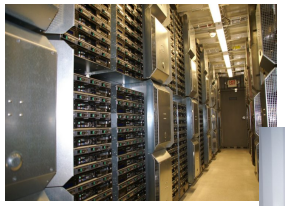


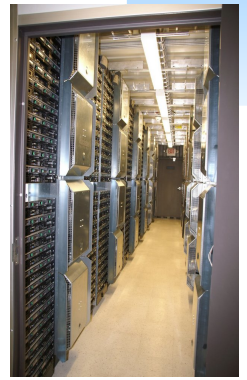
ここ数年で計算システムは大きく変わっている

いまどきの計算システム(Google)

コンテナ型データセンター

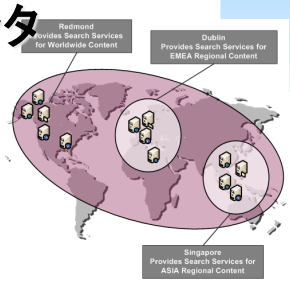
- 物流用コンテナにサーバ格納
  - コンテナー個に1000台以上の
- コンテナに刺さっているホースは冷水を引き入れと、サーバーの冷却に利用した後の温水の排出





Rackableの工場

Microsoftのデータセンタ



- 建屋データセンター
  - 30十万台程度（推測）
  - 管理者1人で5000台のサーバを管理
- コンテナー形データセンター
  - サーバをコンテナに内蔵、電源や水冷機構もモジュール化
  - 徹底的な管理コストの低減
  - 各サーバの詳細は不明

世界のサーバの20%は4社が所有

- 2009年3月6日 Financial Timesの記事から
- 「世界で販売されたサーバの20%は、Google、Microsoft、Yahoo、Amazonが購入している」
- 備考:2008年の日本のサーバ集荷数は553万台

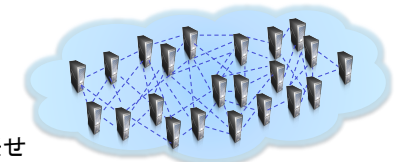
> According to Microsoft research chief Rick Rashid, around 20 per cent of all the servers sold around the world each year are now being bought by a small handful of internet companies - he named Microsoft, Google, Yahoo and Amazon. That is an amazing statistic, and certainly not one I'd heard before. And this is before cloud computing has really caught on in a big way.

大規模計算センター

- データセンターは「規模の経済」

	中規模(1000台クラス)	大規模(5万台クラス)	倍率
通信コスト（1Mbpsの通信回線あたり）	月額95ドル	月額13ドル	7.1倍
ストレージコスト（1GBの容量あたり）	月額2.2ドル	月額0.40ドル	5.7倍
管理コスト（一管理者による管理台数）	140台	1000台以上	7.1倍

- 多数のサーバ
- 従量制(Pay-per-use)
- 事実上、無限大の計算リソース
- バックアップや運用はインフラ任せ



10万台から100万台のサーバ

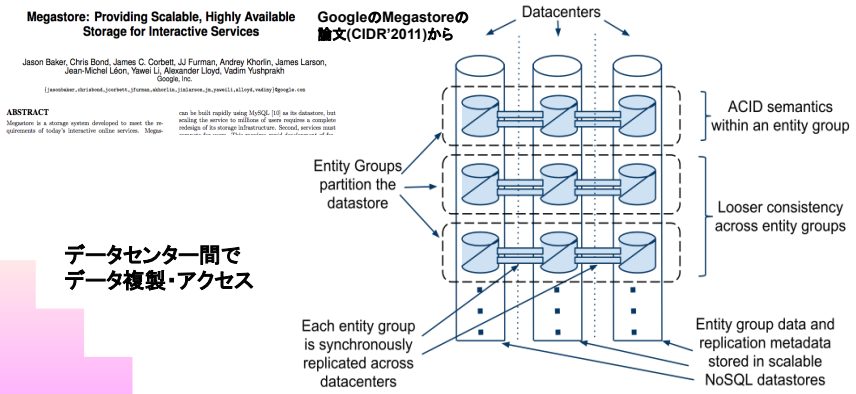
Ichiro Satoh

データセンター(DC)のコモディティ化

- クラウド向けDCの技術トレンド
  - 複製データを複数DCに配置し、性能及び信頼性向上

Megastore: Providing Scalable, Highly Available Storage for Interactive Services

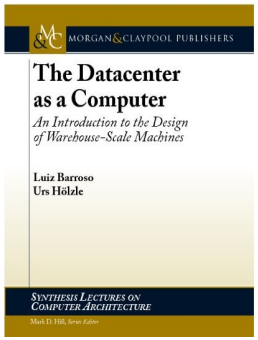
GoogleのMegastoreの論文(CIDR'2011)から



データセンター間でデータ複製・アクセス

データセンター(DC)のコモディティ化

- いまどきのクラウド向けインフラのアーキテクチャ
  - 一つのDCを一つのコンピュータとして扱う
  - DCというコンピュータの分散システムとしてインフラを構築
- コンピュータ(サーバを含む)はコモディティ
  - DCは他社との差別化要因ではない
  - Google、Amazon、Microsoftなどが自前DCではなく、他社のDCを利用することも想定すべき

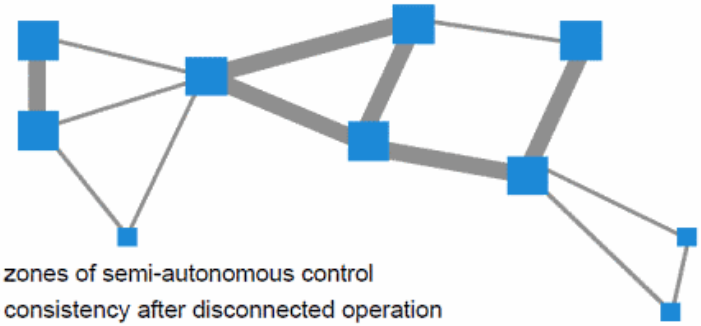


2009年5月に発表

Ichiro Satoh

Design Goals for Spanner

- Future scale: ~10^6 to 10^7 machines, ~10^{13} directories, ~10^{18} bytes of storage, spread at 100s to 1000s of locations around the world, ~10^9 client machines



– zones of semi-autonomous control

– consistency after disconnected operation

– users specify high-level desires:

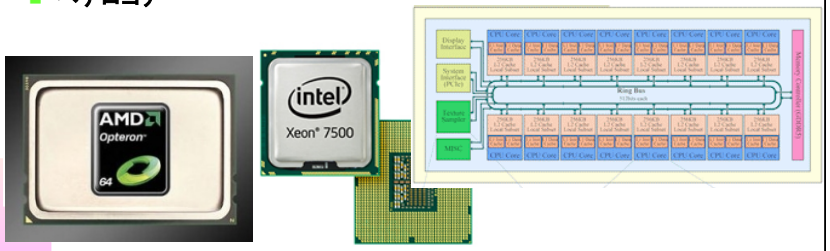
"99%ile latency for accessing this data should be <50ms"

"Store this data on at least 2 disks in EU, 2 in U.S. & 1 in Asia"

Google

メニーコアプロセッサ

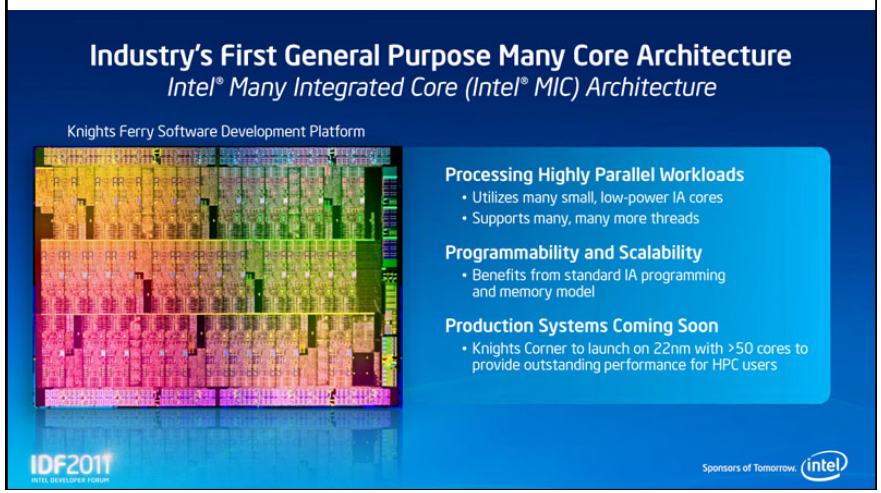
- マルチコアプロセッサはあたりまえ
  - Intelは研究開発用に48コア
  - ノートPCでも2コアまたは4コアはあたりまえ
- CPU+GPUアーキテクチャ
  - Intel Larrabeeでは汎用コアとGPUをリングバス結合
- ヘテロコア



Ichiro Satoh

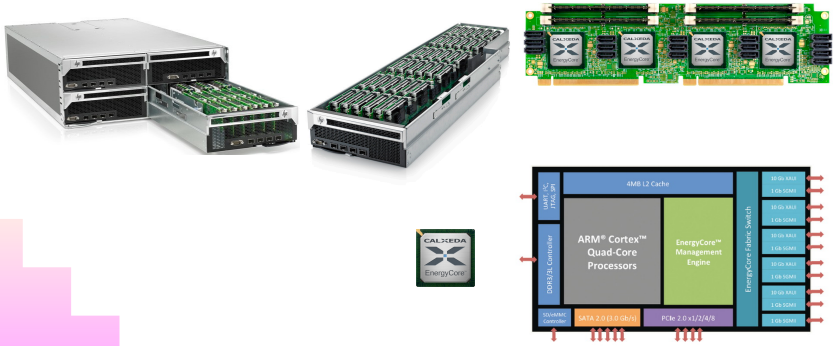
Intel's Many-core processors

Intel's Many-core processors



ARMサーバ

- 組込系プロセッサ(ARM)をサーバに利用
- 消費電力当たりの計算性能
  - Xeonサーバに対して消費電力で89%、スペースで94%(メーカ曰く)



マルチコアARM

- 低消費電力プロセッサ
  - Intel Atom
  - ARM
- 消費電力あたりの計算性能はARMの方が有利

PC/サーバ用

?

携帯・組込用

Ichiro Satoh

いまどきの計算

- 超大規模な分散コンピューティングシステムがあたりまえ
  - 多数のコンピュータで分散処理
  - 計算よりも問題分割が重要
- プロセッサはマルチコア以上があたりまえ
  - 並列計算機の方が普通
  - 力業的な計算でもなんとかなる
- ネットワーク接続はあたりまえ
  - 計算よりも通信
- 低消費電力
  - 消費電力当たりの性能

- 授業では分散システムのための計算モデルを中心にして扱う

Ichiro Satoh

計算モデルの必要性

- 計算機のハードウェア
  - 複雑すぎて扱えない
- CやPascalなどで書かれたプログラム
  - 数学的な厳密さを持つとは限らない
- プログラムを入力から出力を得るものとする
  - 例：数学的な関数
  - ただし、実際に入力を与えても、その答え（出力）が導けるとは限らない

講義の目標

- 各種の計算モデルについて学ぶこと
- 計算の数学的な取り扱いに慣れること
- 計算モデルを通じて計算という概念を理解すること
- ものの原理を見抜くセンスを養うこと

授業資料のダウンロード

http://research.nii.ac.jp/~ichiro/lecture/model2012/

といいつつ、今日の資料は未アップロード

履修者への質問

- 講義内容とレベルは履修者の背景知識と関心により決める
- 履修者の背景知識に関する質問
- 集合または代数論の講義の履修（有／無）
- オートマトンの講義の履修（有／無）
- 一階述語論理の講義の履修（有／無）
- 計算モデルに関する講義の履修（有／無）

計算モデルとは

国立情報学研究所

佐藤一郎

E-mail: ichiro@nii.ac.jp

計算モデル

- 計算モデルの必要性
- 世の中の情報処理システムは非常に複雑
- この情報処理システムがどのような原理に基づくかを知るには数学的基盤を持つ抽象的な枠組みを導入する必要がある。
  - 情報処理システムの抽象的な枠組み＝計算モデル
- 計算モデルを通じて「計算」という概念を明確化する

「計算」とは

- 「計算」とは、
  - ある「アルゴリズム」に基づいて、ある基本的な手順を実行し目的とする結果を得る作業
- しかし、実際の情報処理システムは複雑であり、本来の基本原理である「計算」が不明確
- 数学的な裏付けをもつ抽象的な枠組み（＝計算モデル）を導入し、
- 記号の書き換えとして計算を再現する

計算モデルの必要性

- 計算機のハードウェア
  - 複雑すぎて扱えない
- CやPascalなどで書かれたプログラム
  - 数学的な厳密さを持つとは限らない
- プログラムを入力から出力を得るものとする
  - 例：数学的な関数
  - ただし、実際に入力を与えても、その答え（出力）が導けるとは限らない

代表的な計算モデル

- (逐次)計算モデル:
- 抽象機械計算モデル
- 関数型計算モデル
- 論理型計算モデル
- 項書き換え型計算モデル

- (並列分散)計算モデル
- 抽象機械計算モデル
- プロセス代数(プロセスカルキュラス)
- イベント機械モデル(ペトリネット)

自然現象と計算

- 自然現象を計算として捉える
- 既存手法ではえられない知見をえることがある

- 自然現象を計算機として捉える
- 一般のコンピュータよりも高性能なこともある。
  - 例:DNAコンピューティング

- 自然現象が計算システムのヒントになることがある

宇宙はコンピューターだ——と言われてもピンとこないかもしれません。でも本当なんですよ。プログラムは「物理法則」。計算しているのは「自らの構造」。かの天才・ホーキングが自説を修正したように，何でも吸い込むブラックホールでさえ，ちゃんと計算結果を出力しているのです

例:DNA計算(DNA computing)

- Leonard Adleman (1994)
- 分子生物学的な実験手法による有向ハミルトン経路問題の解法を発表
- 経路問題をDNAの塩基配列として表現
- 分子化学的操作のみによって変化(計算),
- 大量の分子が並列的に化学反応をおきる
  →超並列性

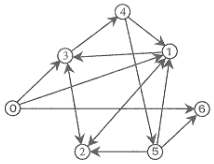


図1　有向グラフ

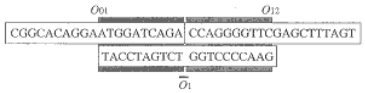


図2　経路 0 → 1 → 2 の DNA 表現

例:Ambient Calculus

- 細胞(膜)の変異を表す計算モデル
- 構文

$$X ::= \diamond \mid X \circ X \mid (Y) \quad \text{(cyto brackets)}$$
$$Y ::= \bullet \mid Y \bullet Y \mid [X] \quad \text{(exo brackets)}$$

- 公理

$$\diamond = (\bullet) \qquad \bullet = [\diamond]$$
$$X \circ (Y) = (X \bullet Y) \qquad [X] \bullet Y = [X \circ (Y)]$$

- 推論

$$(Y)\,(Y') = ((Y))\,Y' = ((\diamond)\,(Y))\,Y' = ((\diamond)\,Y\,Y') = \diamond\,(Y\,Y') = (Y\,Y')$$

$$X \circ (Y) = (X \bullet Y)$$