# 並列オブジェクト, Twitter, 新型ウイルス抗体解析など

米澤　明憲
東京大学情報理工学系研究科
同　情報基盤センター

# あらすじ

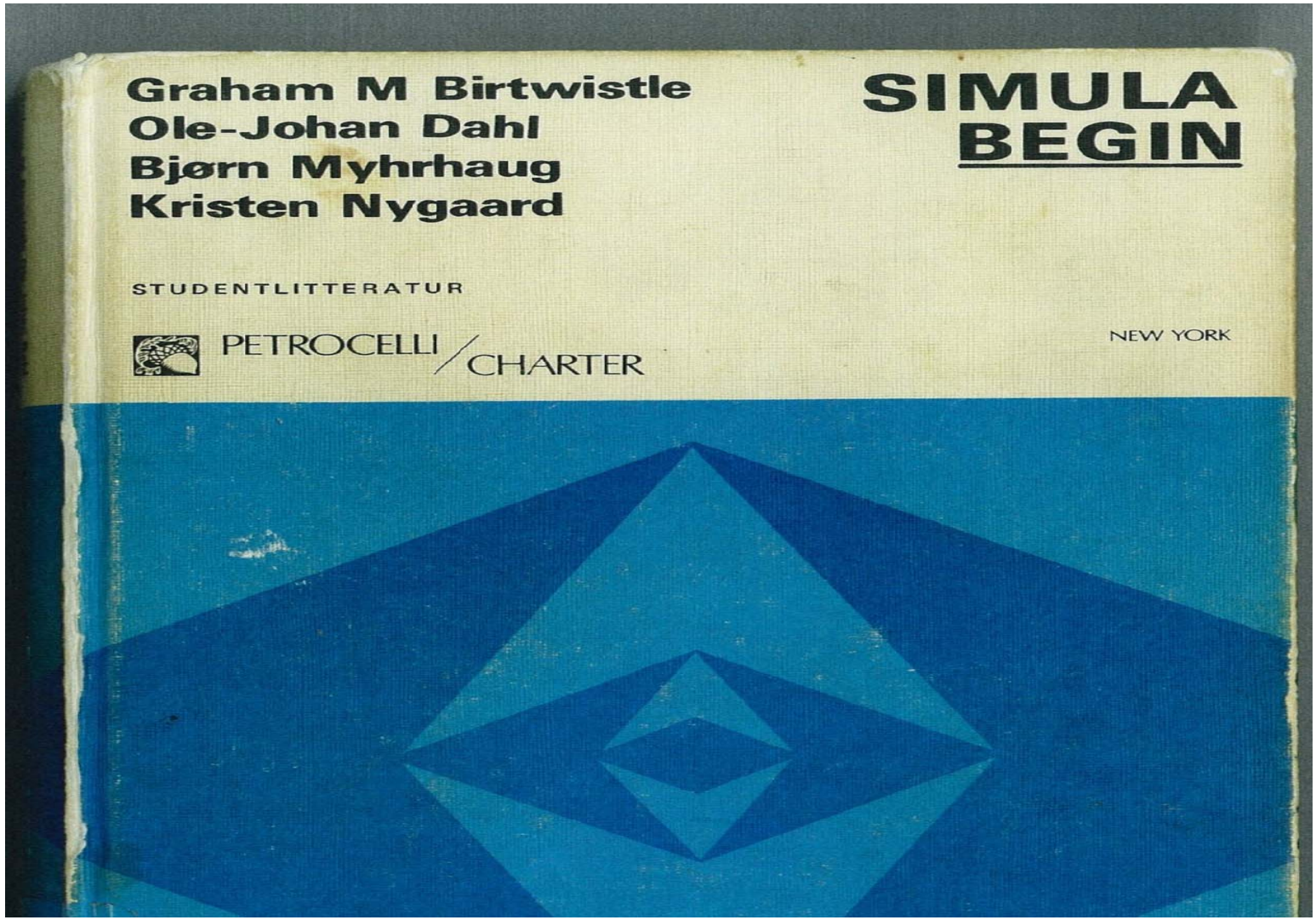
1. 並列オブジェクトとは
2. 並列オブジェクトによる大規模システム
   - **Second Life**
   - **Twitter（Facebook）**
   - **NAMD (ナノバイオ分子動力学）**
3. 我々の貢献
4. 超並列、many-core時代に入って
5. おわりに

# 並列オブジェクトとは？

1970年代半ばから

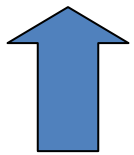
まずは、オブジェクトとは？

# ダール・ニゴールさんのSimula67の教科書 [1973]



# Simula67後, そして70年代前半

- Smalltalk – 1972 Alan Kay, Language interface to DYNABOOK
- CLU (abstract data types) – 1973 B. Liskov
- Frame System – 1974 M. Minsky
- Actor – 1973 C. Hewitt, Universal modular forms for AI
- Capability-based OS – 1975 W.Wulf
- Entity-Relationship Model – 1973 C. Chen, data model



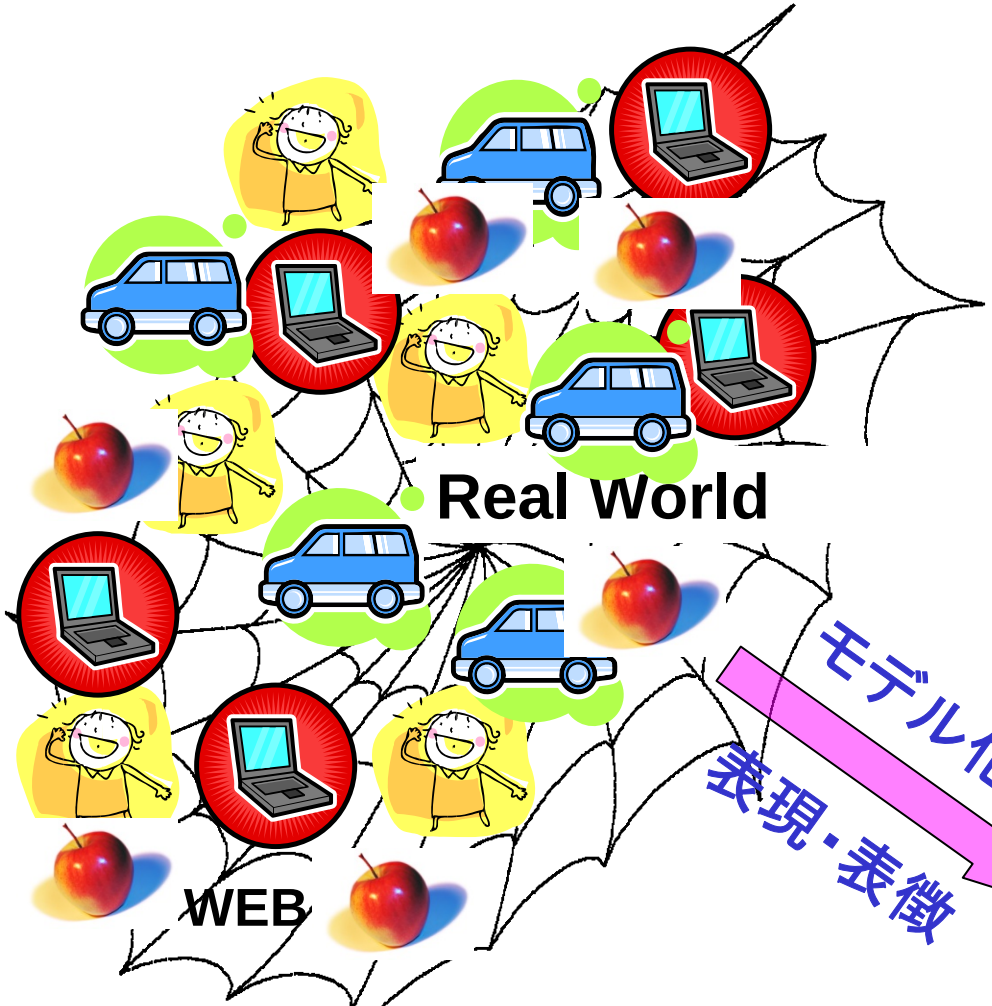
「プログラム表現」や「知識表現」の形式を
いかに、構造化・モジュール化にするかの探求

私はモデル記述とプログラミングの両方を考えてゆきたいと思った！

1. 「世界」のモデルをつくり、それに従って
   「世界」をコンピュータの上でシミュレートする

2. より強力なプログラム記述の枠組みを考える

「自然な」
モデルやオブジェクトと
はどんなものか？？



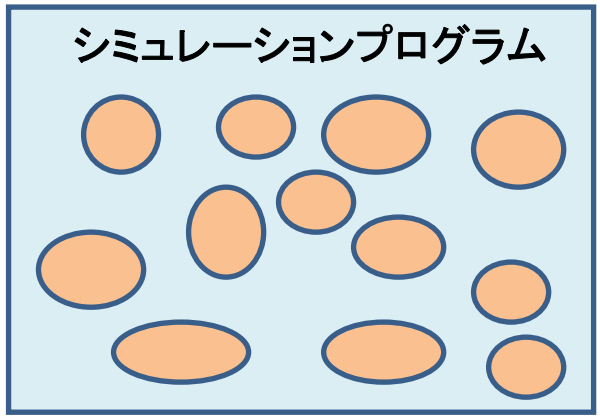

モデル化

表現・表徴

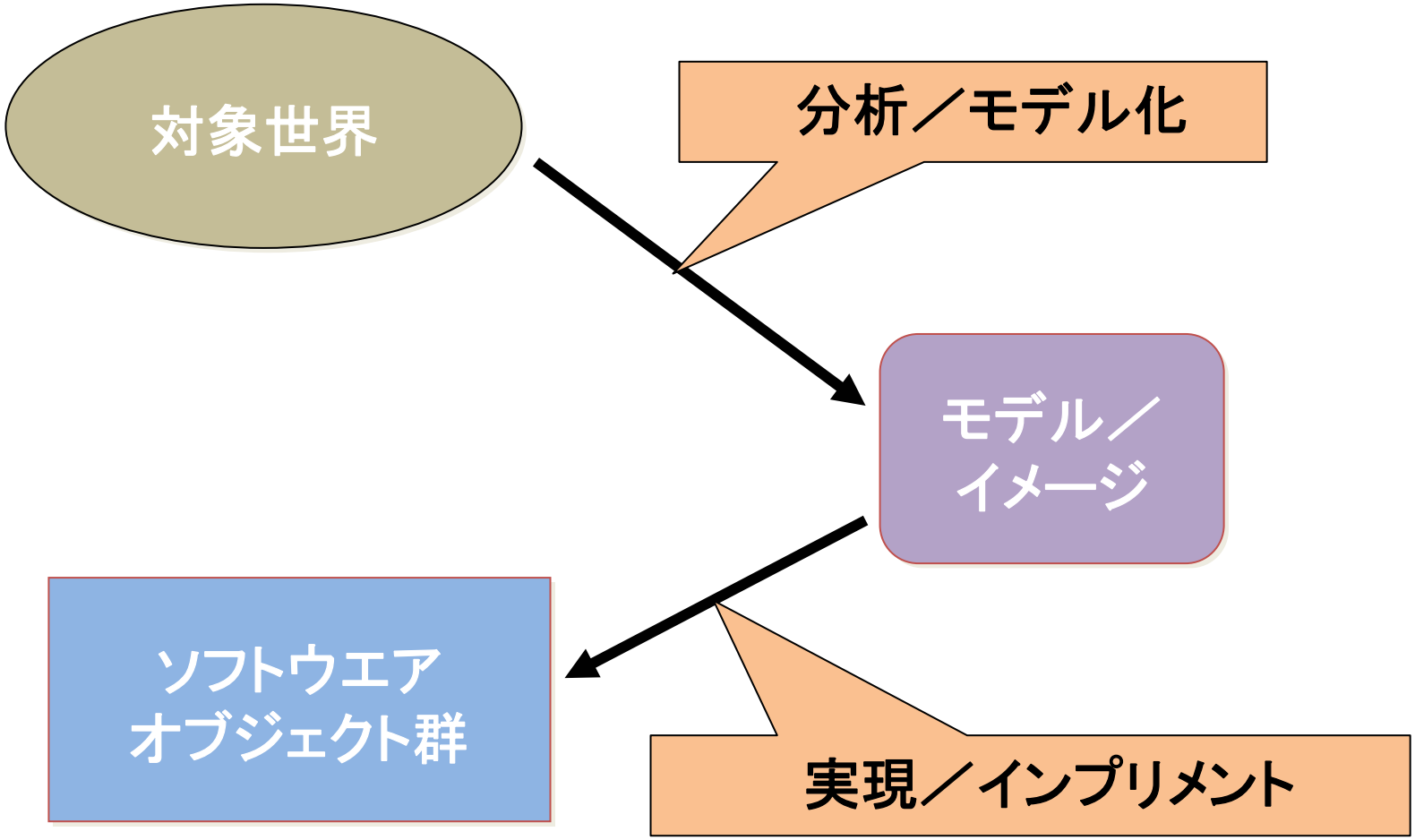
事物，　人々，
コンピュータ群 ＆
それらの相互作用

# モデル化して
# シミュレーション

ソフトウエアモジュール

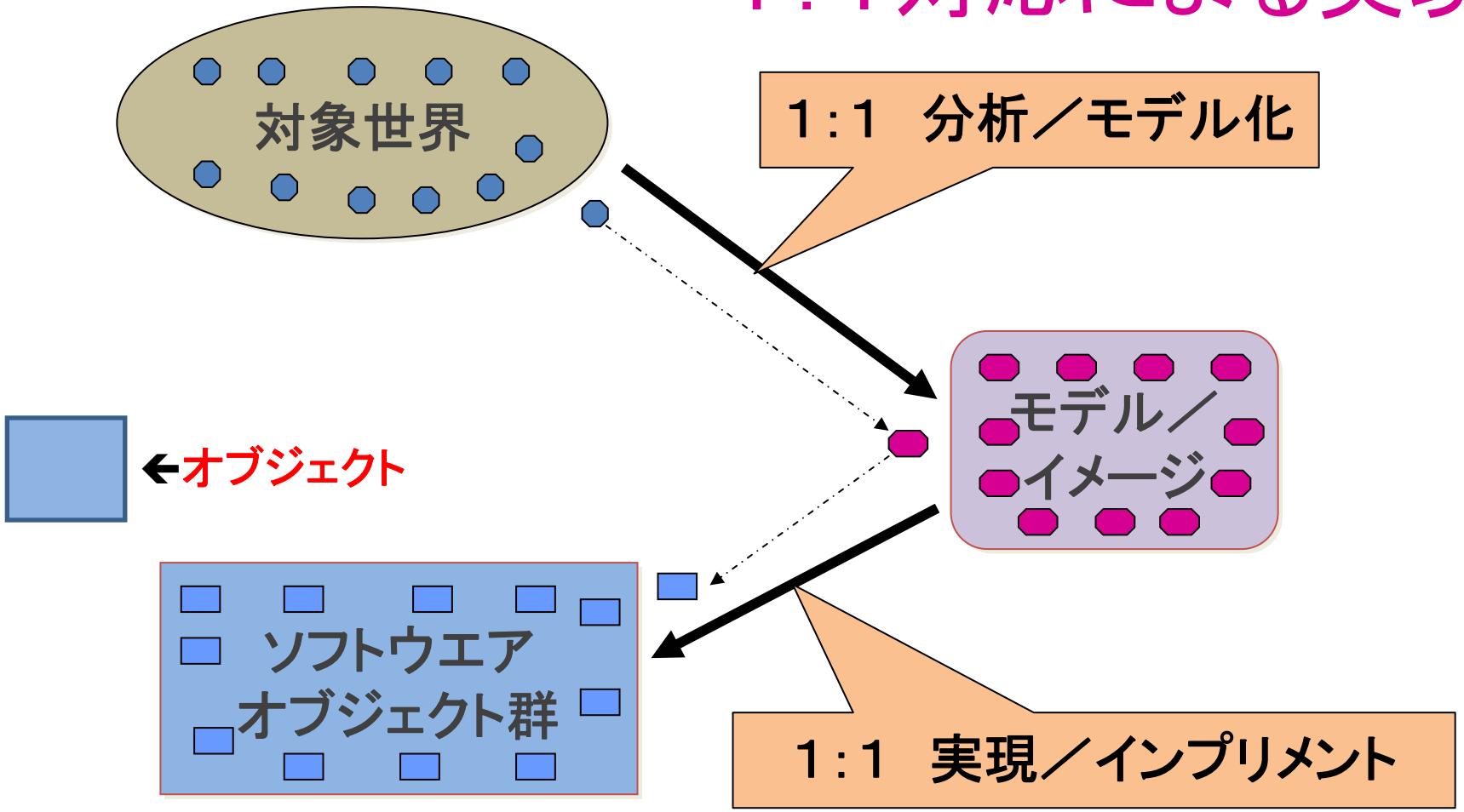


# モデル化とシミュレーション



# 自然／直接的なモデル化と実現

## 1:1対応による実現

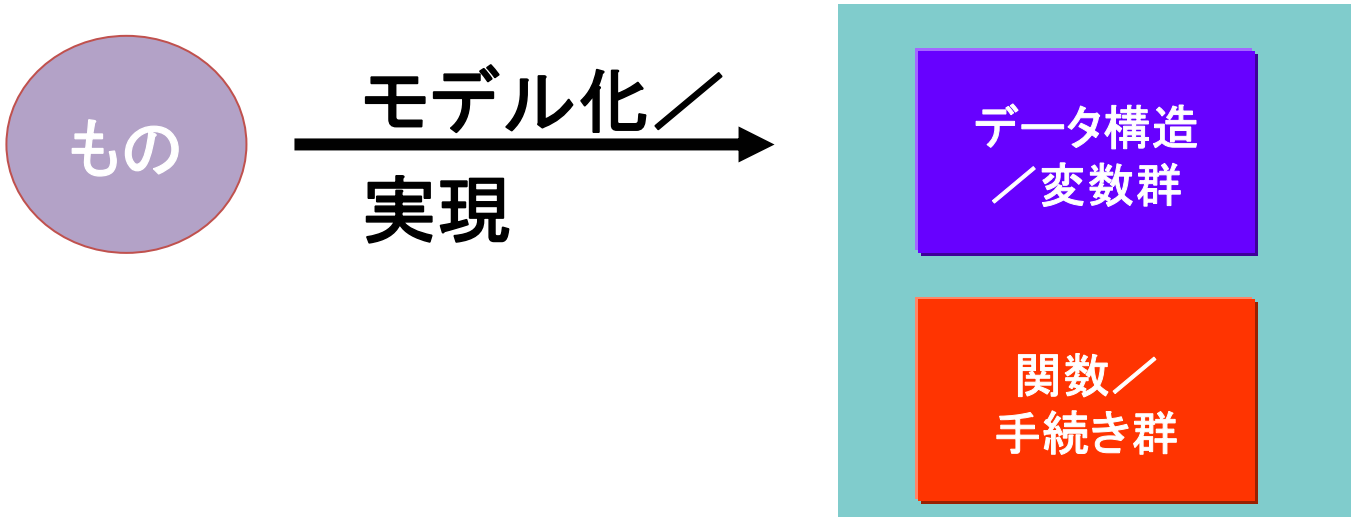


# 「もの」を「オブジェクト」として表現・実現する

- 「もの」の状態や記憶を
  → データ構造（変数群）として
- 「もの」の機能や性質を
  → 関数・手続群として

そのデータ構造と関数・手続きを
一つのカプセルに封じ込めたソフトウェアモジュール
＝ *「オブジェクト」*
（データとそれを解釈・利用する手続きを合体）

# オブジェクトとは




- オブジェクトとは、
「もの」をモデル化する
データ構造とそれ利用する関数・手続を
1つのカプセルに入れて合体した
ソフトウエアモジュール

で、並列オブジェクトとは？

# この時代は

- ICの時代が過ぎ。。。
- ARPA NETが拡大中
- VLSIが簡易に設計できる見込みがついた！
  - C. Mead & L. Conway, *Intro. VLSI Design*, 1979

- 将来、あり余るマイクロプロセッサが安価に使えるようになるだろう。。。

# 世界の並列性も自然に表現できる！

- 雨は並列に降る、

- 「もの」は並列に存在する
- 「もの」は移動する
- 「もの」同士のインタラクションや作用は？？

**並列オブジェクト(Concurrent Object)**

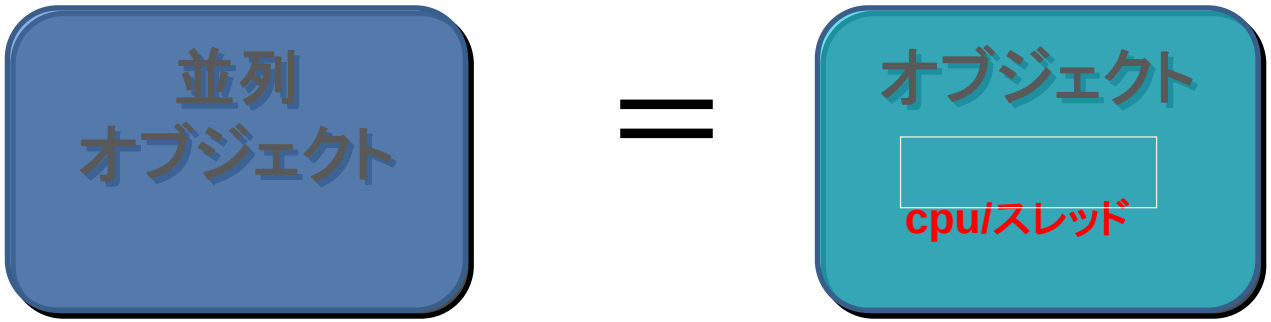
あるいは **“stateful actor”**

# 1975年ごろ並列オブジェクトを思いついた

モデル記述とプログラミングの両方を目指して,…

- 並列オブジェクト:

1. (状態をもつオブジェクト + スレッド)をカプセルに閉じ込める

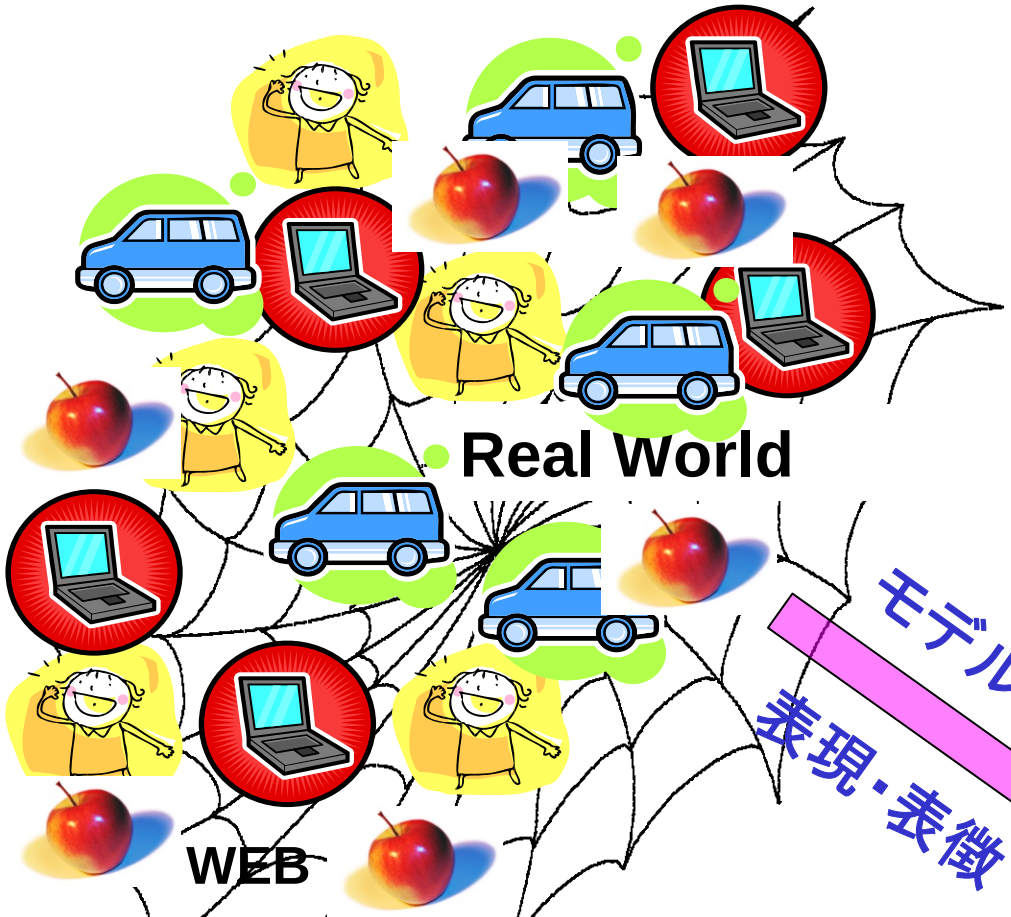



2. 並列オブジェクト同士の非同期的メッセージ送信

- 少し違ったアプローチ(Actor)

-C.Hewitt and H.Baker:
*Laws for communicating Parallel Processes*, IFIP1977
-G.Agha: *A Model for Concurrent Computation in Distributed Systems*,
MIT Press 1987

# 並列オブジェクトによるモデル化とシミュレーション

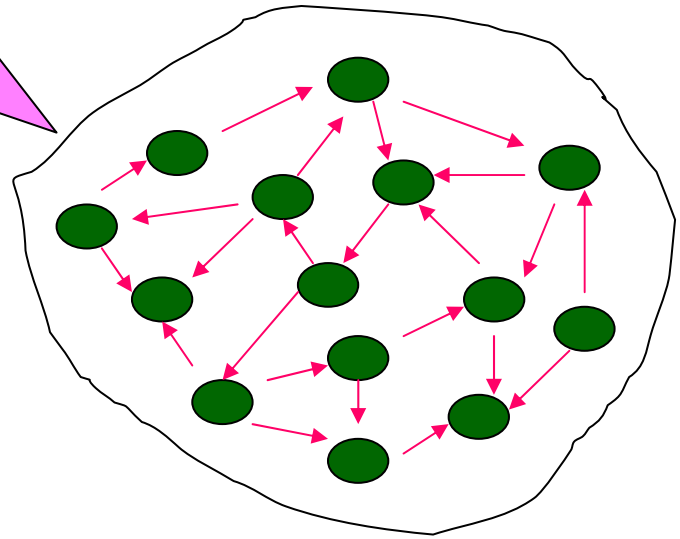



事物，　人々，　コンピュータ群＆　それらの相互作用

モデル化

表現・表徴

並列オブジェクト群 &
メッセージのやりとり



# 並列オブジェクトのポイント

1. メッセージ(イベント)駆動型処理(message-driven)
2. 非同期メッセージ送信(asynchronous comm.)

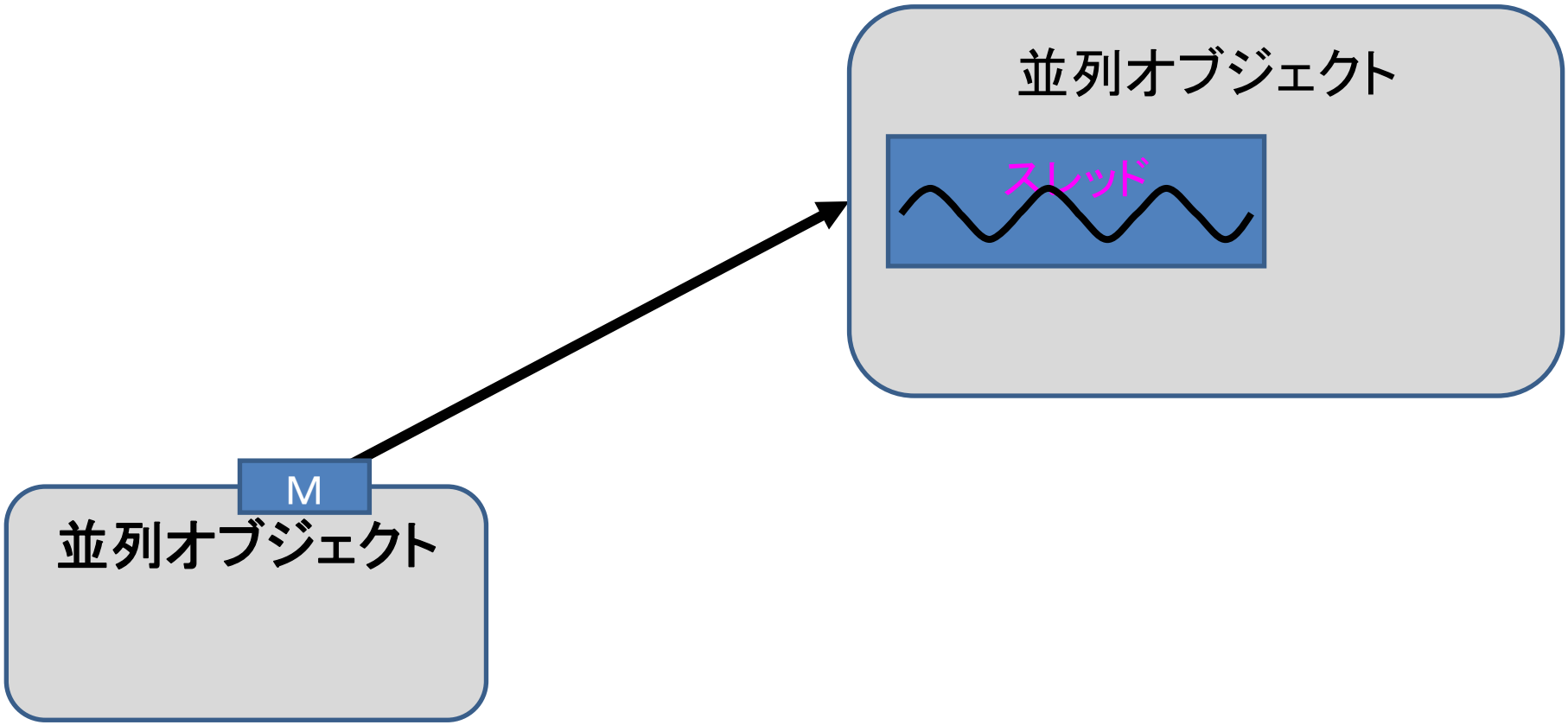
3. 情報の交換・共有手段はメッセージのやりとりのみ

4. 制御の流れとデータ・情報の流れが一致してる

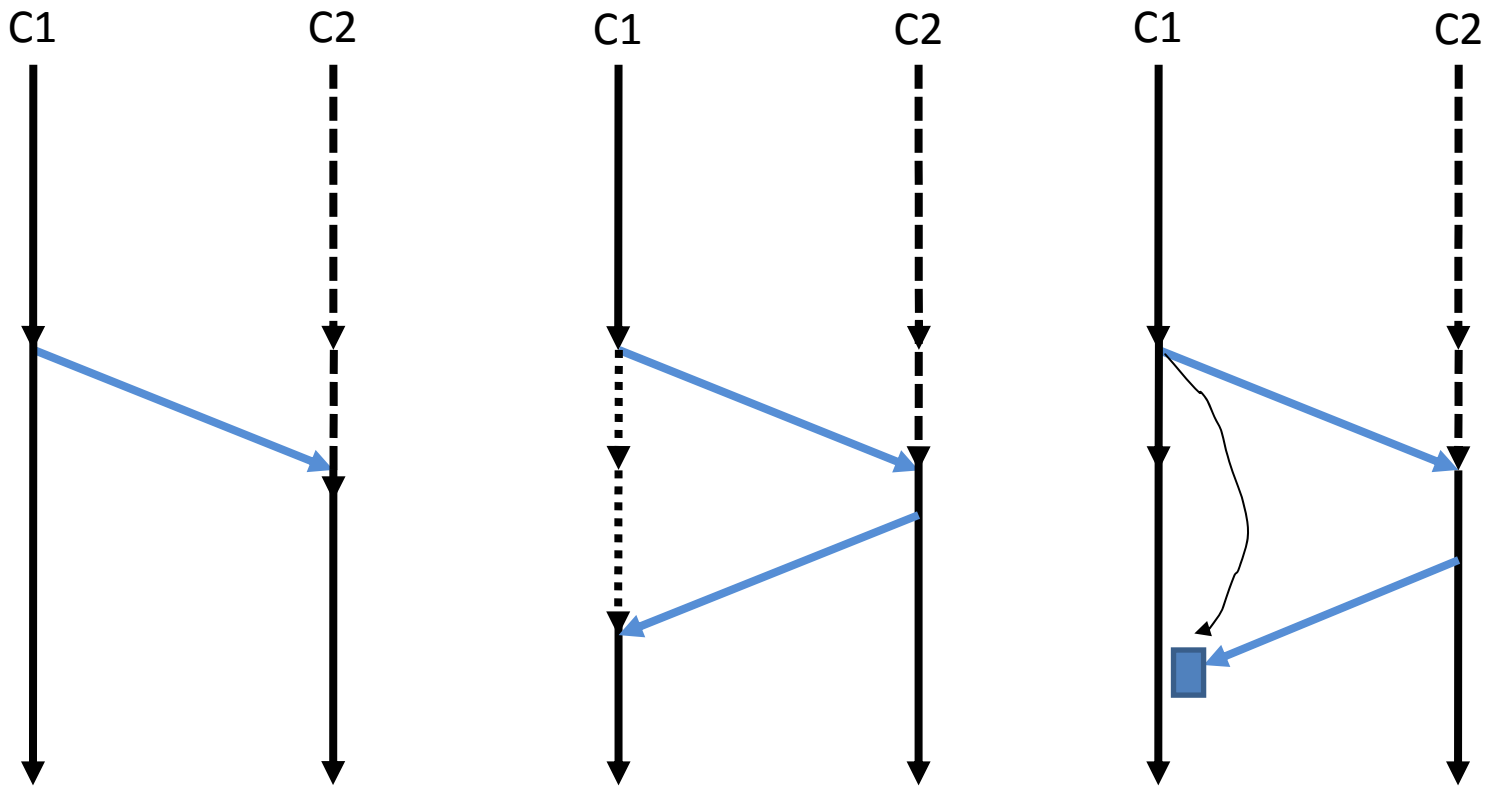
5. 多数の並列オブジェクトを用いて(超)高度並列性
6. 安全な軽量プロセス(light-weight process)
   - lock, unlock操作が不要

7. 自然なモデリング
   自然な分割の単位　ー　機能(データとプログラム)

# 1. メッセージ駆動



# 2. 非同期なメッセージ送信

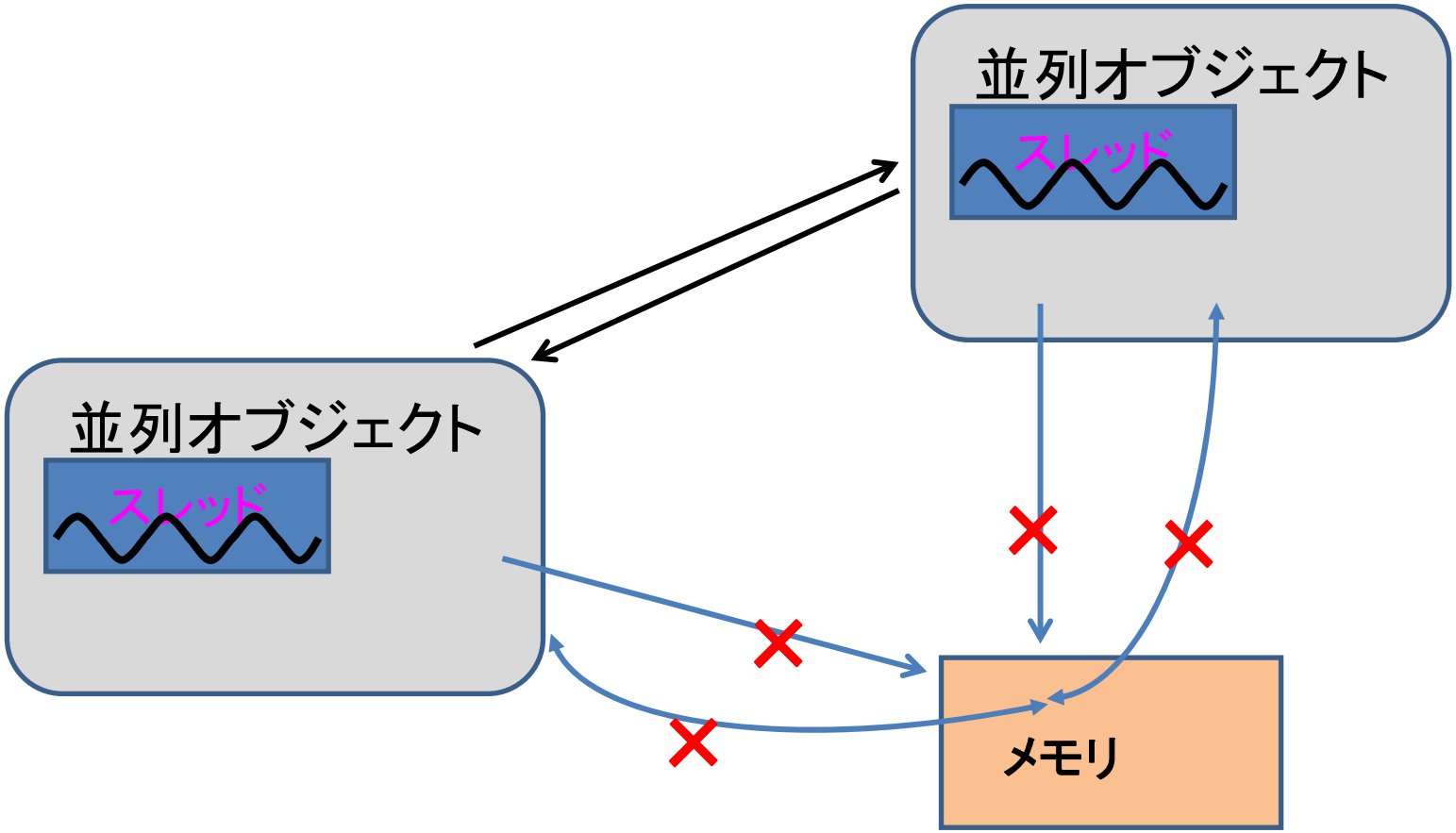



型1

型2

型3(future)

# 3. 情報共有・交換手段は
## message passing のみ

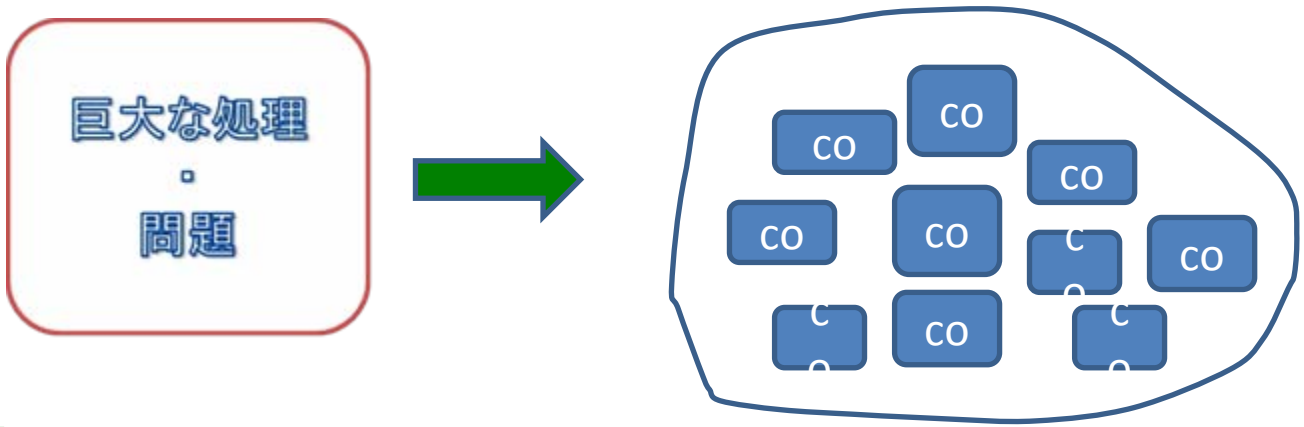


# COでモデル・表現できるもの

- 粒子（物質）、構造物
- 人、エージェント

。。。　もっと抽象化して

- 仮想プロセッサ（VP）/軽量スレッド
- 問題を分割して得られる多数の細い処理・部分




  - 分割して得られる部分はCO
  - この部分は計算と通信を行う

# 並列オブジェクトの利点と大規模応用

1. 自然なモデリングの表現

Second Life で分かりやすく強力な記述能力を発揮

2. 自然で安全・高速な並列・並行プログラミング

Twitter, Facebok

3. 高性能で高生産性な並列・並行プログラミング

Charm++ → NAMD → Swine Fluの抗体解析

# Second Life

# Second Lifeシステム

- Linden 社が、オンライン仮想世界を
  ユーザが作る仕組みを提供している、
  ユーザに自由に作らせている
- 数100万人のユーザがいた。
  - 教育用
  - ビジネス（営業、プレゼン、紹介、。。。）
  - ゲームをつくる
  - 。。。

◎　登場物やアバタは並列オブジェクトとして
表現・プログラムされている

◎　Second Lifeで、これらはscripted objectと呼ばれる

Second life demonstration

# 珍しい使い方

- 仮想世界でアバターを用いてアンケート調査を行って、現実世界でのマーケティング
- セカンドライフ内の仮想店舗で接客を行うアバターを操作するオペレータの人材派遣を行うサービス
- アメリカ議会では仮想世界での経済活動について課税するといった議論も

# 「セカンドライフ」で使われる並列オブジェクト


○登場物やアバタは
並列オブジェクトとして
表現・プログラムされてる!!

**Image from “Programming Second Life with the Linden Scripting Language” by Jeff Heaton (http://www.devx.com/opensource/Article/33905)**

# Second Life の中の並列オブジェクト

- Jim Purbrick, Mark LentcznerはOOPSLA07の招待講演：
  *“Second Life: The World’s Biggest Programming Environment”,*
  で、つぎのように述べている:

  ＃　登場物とアバタは互いにメッセージを
  やりとりをしながら協調している。

  ＃各登場物とアバタは以下のようにプログラムされる:

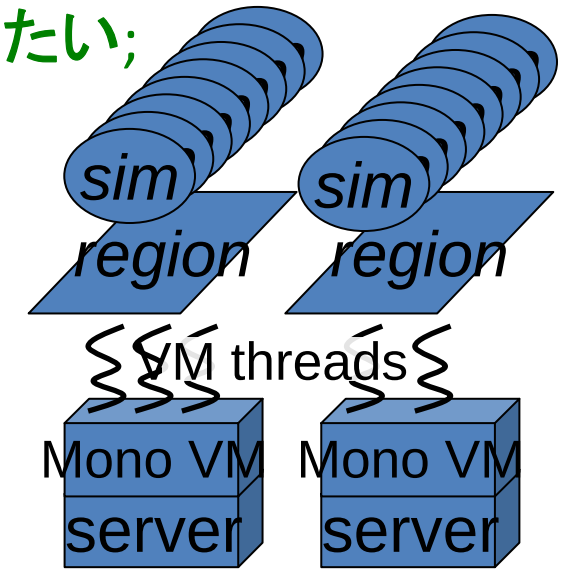
  1. 自分自身の状態をもつ,
  2. 到着するメッセージに反応する手続き・ソッドを持っている、
  3. 異なる状態ごとに異なる反応をすることができる、
  4. 自分自身のスレッドをもっている

  並列オブジェクトそのもの

  – Second Lifeでは、約200万個の登場物がプログラムされ、それらが動いている。

# Second Lifeの実装

- より多数の“sim (simulated objects/scripted objects/Cos)” を動かしたい
  - **1つのregion（小世界）で、**
    **常に1000個もの**scripted object/sim object
    **すなわち、並列オブジェクトを動かしたい;**
- 実行されているSimのregion間に
  自立的移動 を高速にしたい




***Mono: MS CLIと互換**
**オープンソースの実行時系**

Second Life Teleporting Demo

# テレポート（4次元での移動）する

# アバターやSimObjectの小世界 (region)間の移動の実装

- 我々のJavaGoX [ASA/MA'00]
  - Javaのバイトコードを変換する手法をベースにしている。これによって、現在JVM上で実行中のオブジェクトを容易にかつ効率よく移動・移民させることができる。
  - Second Life では我々の方法を用いて、Mono ベースのスクリプト実行マシン上で、移動・移民を実現している。
  - "regions"間の移動・移民
    - region（小世界）群は

      複数のサーバーで管理・実行されている
    - アバタなどの移動は、別のサーバによる実行となる


Image from "EVOLVING NEMO" in New World Notes at Second Life Blog (http://secondlife.blogs.com/nwn/2005/06/evolving_nemo.html)

# Second Lifeでなぜ並列オブジェクトが使われたか？

- 並列オブジェクトが仮想世界の登場物を
  直接的にシミュレーション・プログラミングできる！
  - これによって、
    モデル化が容易になり
    かつ並列プログラミングが「安全」で「やさしく」なる

# Twitter

# Twitterシステム

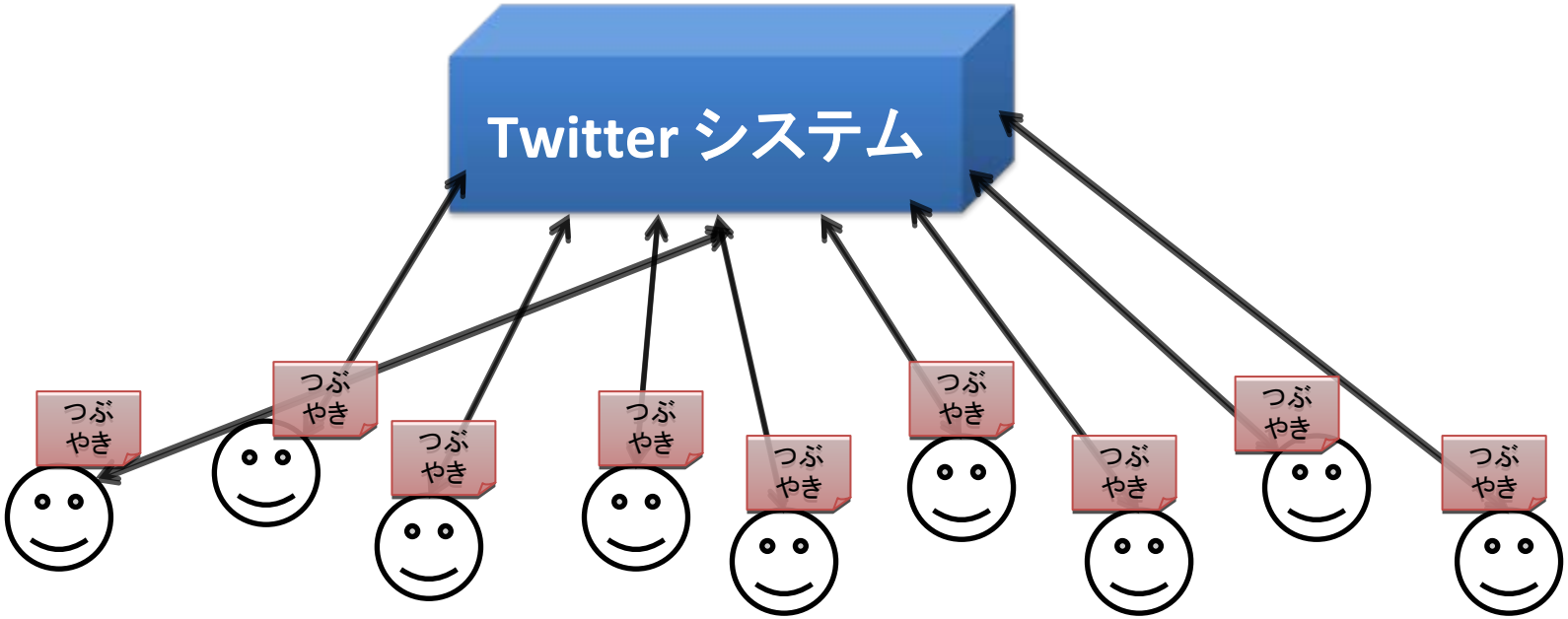
- 米国Twitter Inc.のサービスで、利用者同士が「つぶやき、さえずり」を共有できる
- メールとブログの中間サービス（１４０文字まで）
- 事前に登録した利用者にリアルタイムで伝わる

- 日常の出来事を「つぶやき」として共有で、親睦や迅速な情報伝達が可能
- PC，携帯電話等で使えるため、新しいメディアとして注目を得ている

# 伊藤譲一　twiiter　を語る

# Twitter (ツイッター) とは

- 利用者同士が「つぶやき」と呼ばれる
  短いメッセージを互いに送りあう通信サービス
  - 利用者数は5000万人ともいわれている
    - 米国では多数の政府機関が情報公開等に利用している



# Twitterの重要性

- 利用者数： 5000万人以上 ＝≫ 。。。
- 速報性：
  Obamaの動静、声明
  インフルエンザの最新情報（勧告、感染例、治療法）
  その他、NASA,疾病予防局、FBI、国土安全省、。。。

- 2008年11月8日、ムンバイ同時多発テロ、現地の目撃者や生存者がTwitterを用いて発信。既存メディアに先んじて世界に発信・報道された

# Twitterと並列オブジェクト

- Twitterの実装：

  超大量の「つぶやき」を同時に高速処理する必要

多数の並列オブジェクトによって

•Scala言語とその並列オブジェクト(actor) 機構でプログラム！

古い実装では　Ruby on Railsが使われていた

# なぜ並列オブジェクトが使われたか

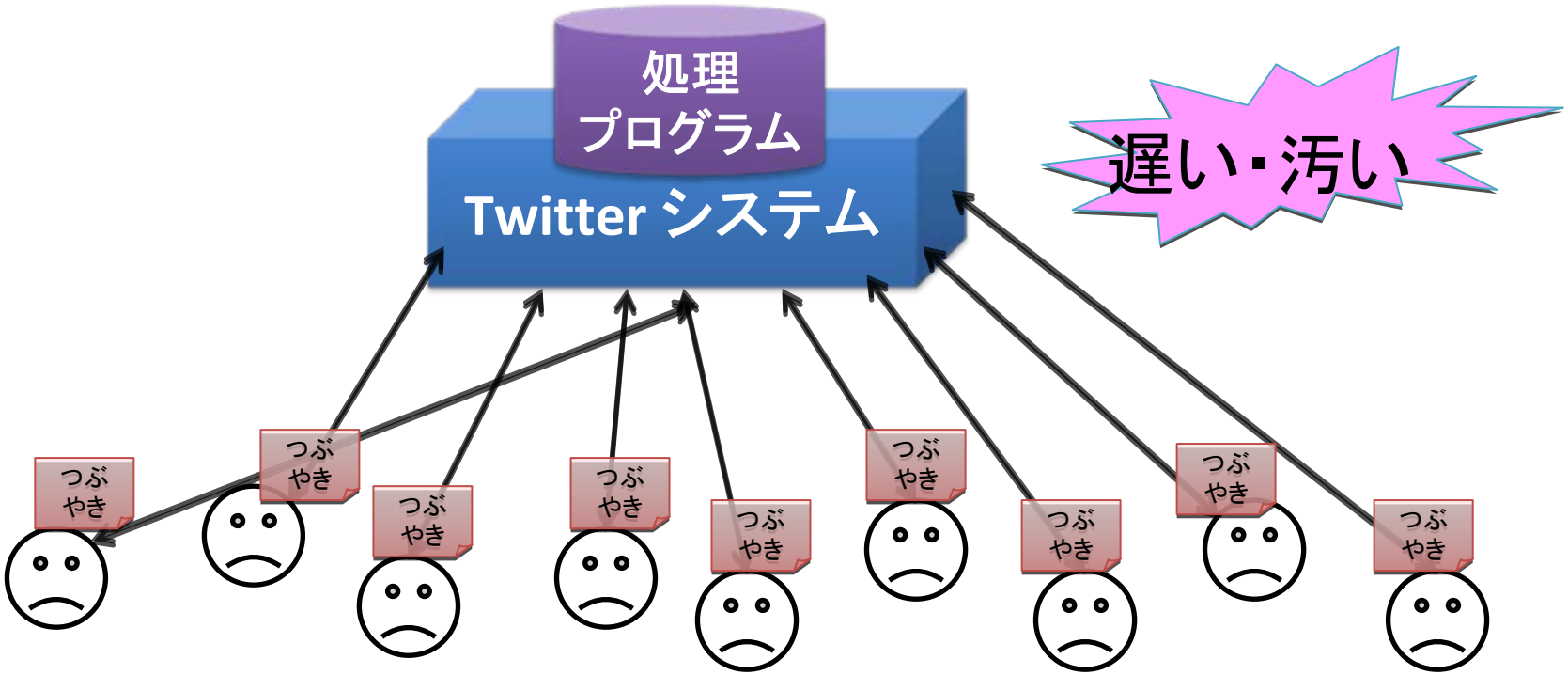
Robey Pointerの話:

ユーザがログインするごとにバックエンドに 並列オブジェクトを生成してゆく

- 並列オブジェクトがプログラムの記述を簡潔にする！
  - メッセージ駆動性と非同期メッセージ
  - コア部分は1600行以下のプログラム
- 高速処理が可能かつ、極端な負荷集中に対処できる
  (瞬間的に数万？の同時つぶやき)
  - 軽量スレッド(と多数の並列オブジェクトによる負荷分散)
- 普通のthreadを多数使って、「つぶやき」の待ち行列の処理は通常の負荷なら可能だが、それでも
  thread同士の排他処理(lock, unlock)が困難！
  プログラムの安全性が保証が困難
- 一つの並列オブジェクトには一threadしかない、lock,unlockが不要

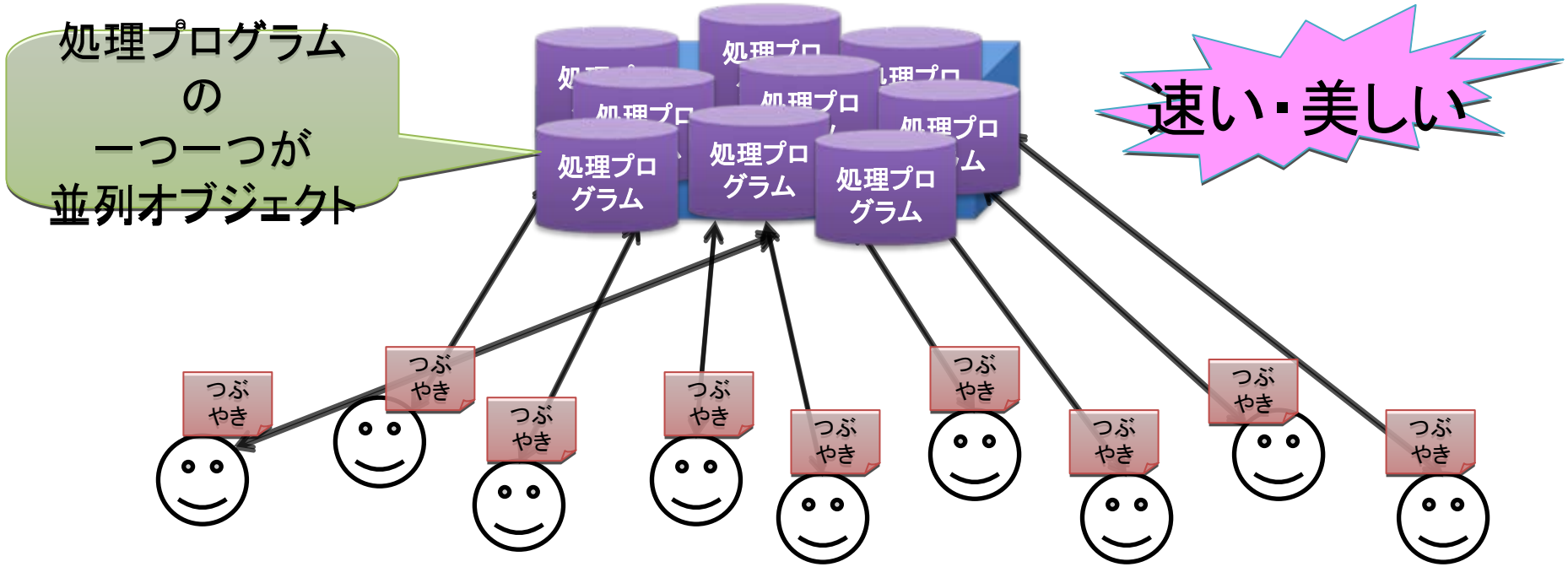
# 並列化を行わずに Twitter を作成したとしたら…

- 「つぶやき」の受信・送信処理を行うプログラムがシステムに一人しかいないようなもの
  - どんなに高速に処理してもさばききれない



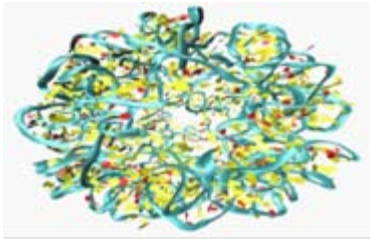
# 実際の Twitter システム

- 「並列オブジェクト」を用いて実装されている
  - このため高性能で安定したシステムの構築が容易になった
    - 具体的には、「つぶやき」の処理を行うプログラムを「並列オブジェクト」として多数用意している



# NAMD

# NAMD[1996]

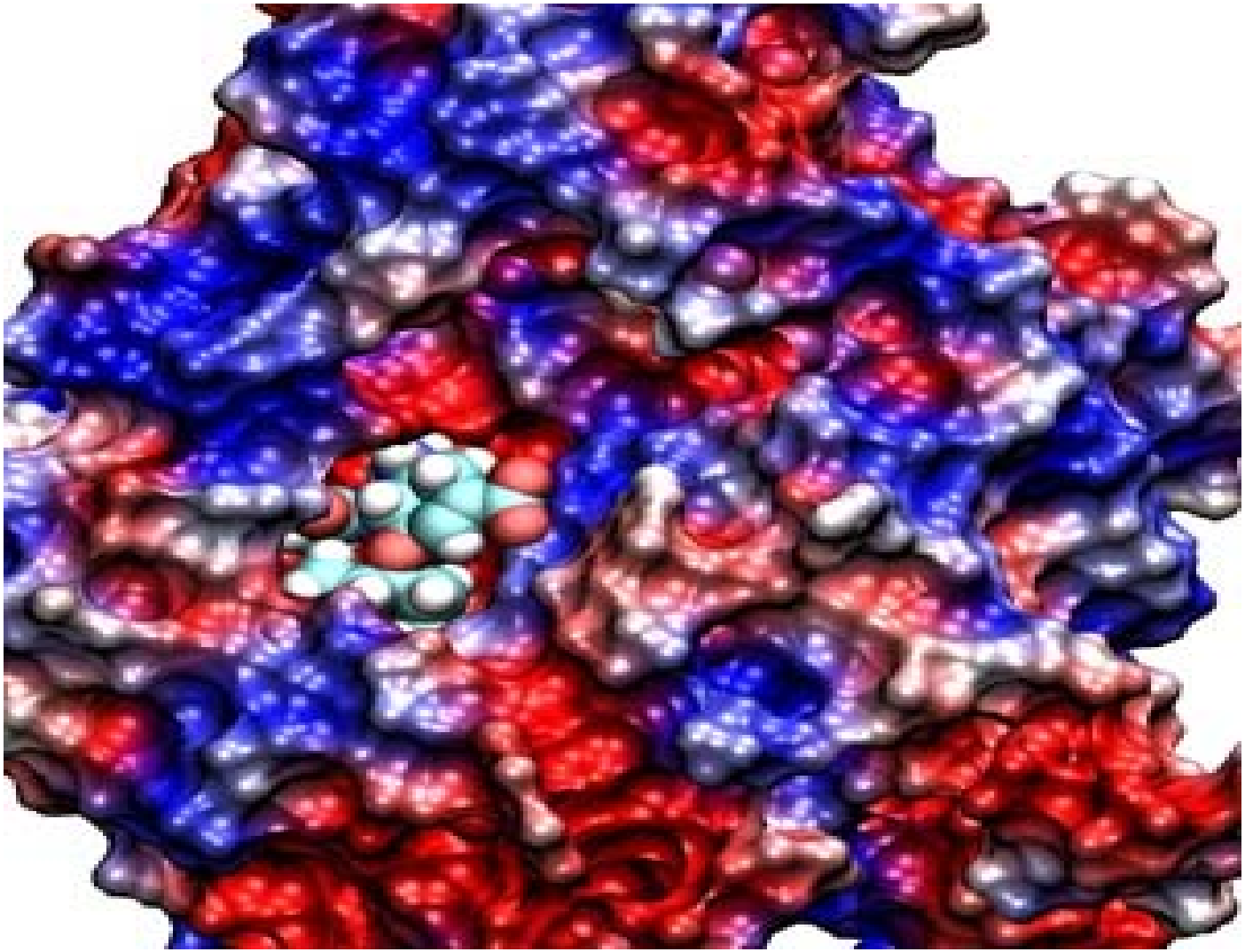


- スパコン向け、ナノスケールの分子動力学simulator
- 2002年、Gordon Bell 賞を受賞
- イリノイ大学の2グループの学際共同開発
  計算機科学のProf. Kale
  理論生物学のProf. Schulten
- 並列オブジェクトシステムCharm++で書かれている

"**NAMD** is a parallel, object-oriented molecular dynamics code designed for high-performance simulation of large biomolecular systems."

- 1 fs 時間を1ステップとしてシミュレーションする
- 各分子にかかる力を1ステップごとに計算、
  次の時刻での各分子の位置と速度を更新

*This still from a Quicktime movie represents a view of the drug buried in the binding pocket of the A/H1N1 neuraminidase protein.*



# Charm++[1993]

- Prof. Kaleのグループが設計・開発した、並列計算機向け、 並列オブジェクト言語システム

  (a concurrent object-oriented system with a clear separation between sequential and parallel objects)

  我々が設計・開発したABCL言語への引用多数
- NAMD, OpenAtom（量子化学）, ChaNCA（銀河生成）等の開発に使用
- 実行時系に動的な負荷分散のframeworkを持つ
- 1千から1万台のCPUをマシン利用して動作している
- Charm++で記述されたアプリケーションは、
  - 米国の二つの主要スーパーコンピュータセンターが保有している計算資源の15% から20% を使用するほど

# Charm++の開発目標

- スパコン向けの「並列オブジェクト」ベースの
  プログラミングシステム　http://charm.cs.uiuc.edu/
- 目標:
  1. 並列アプリケーションの性能を向上させる
  2. プログラマー・プログラミングの生産性を向上させる

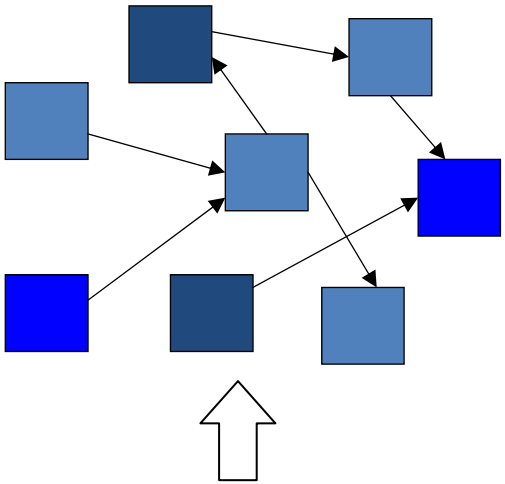
- 利用形態
  - フリーソフトとして流通
  - 複雑で非正規かつ動的なアプリケーションの
    迅速な開発
  - 数千プロセッサー以上のマシンでスケールする

# 大規模アプリでの
# オブジェクトに基づく並列化（仮想化）

基本アイデア：

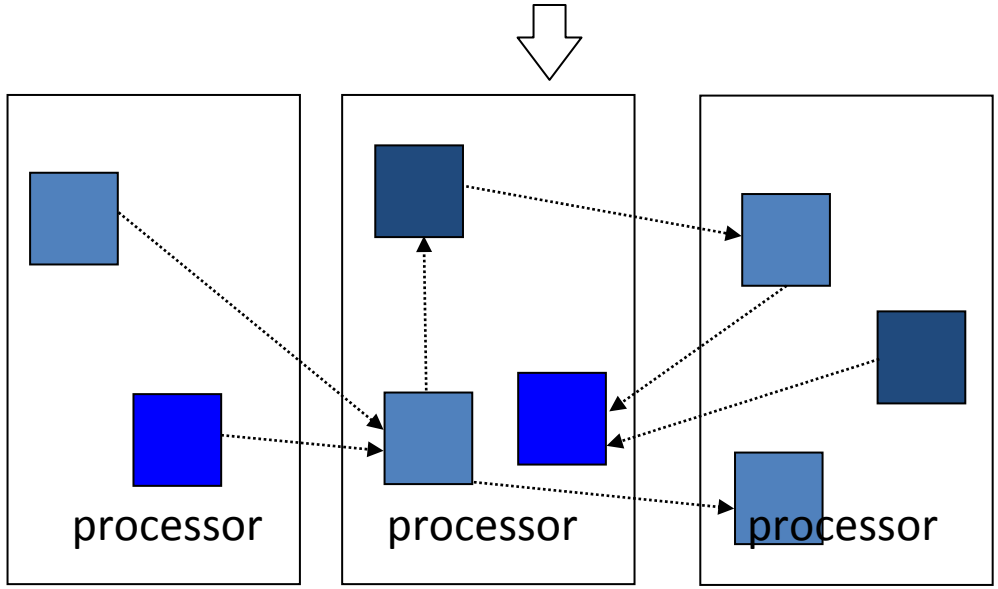
- アプリの全計算（機能）を多数の並列オブジェクトに分割
- システムにその並列オブジェクト群をプロセッサにマップさせる

ユーザの関心はオブジェクト（機能）の間の相互作用のみ！



*ユーザの観点*

*システム実装の観点*




実装上のオブジェクトの配置等はシステムにお願い！

# 並列オブジェクトの移動と負荷分散

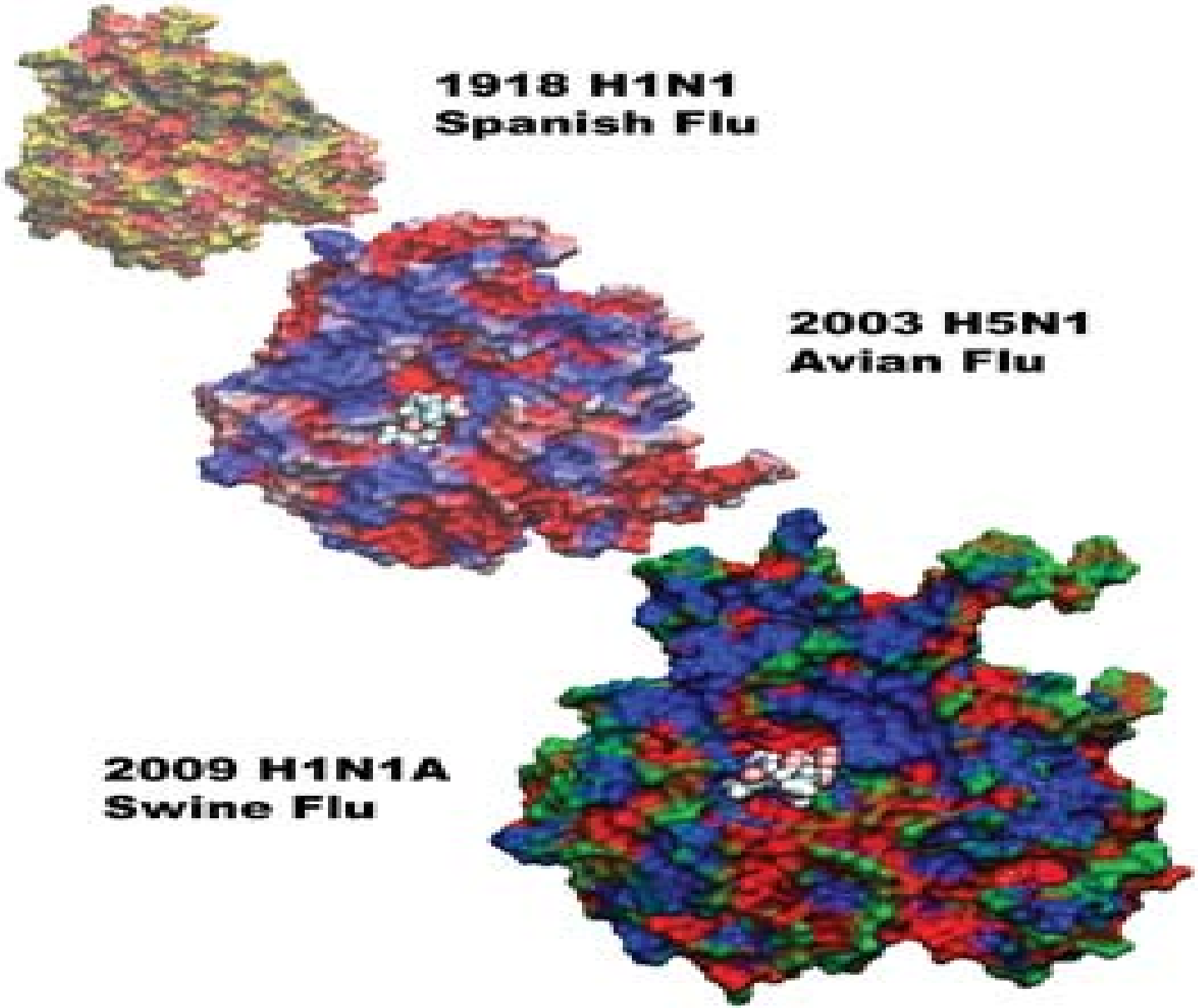
- 仮想プロセッサ（VP）/軽量スレッド
  - 物理プロセッサから出たり・入ったりする

VP（並列オブジェクト）を移動させる

VP（並列オブジェクト）を移動させて
負荷分散・負荷バランス

# NAMDによる新型ウイルスの解析

- イリノイ大のスパコンで新型ウイルス（A/H1N1タンパク質）の分子構造解析をしていた [2008]
- 新型ウイルスで死者が出た [2009.4]
- テキサス大の最新鋭スパコンRangersの2000~
  3000プロセッサを2週間借り切ってシミュレーション
- このシミュレーションで、タミフルがどのように neuraminidase に結合するか明らかにし、
  A/H1N1タンパク質の構造変化（ウイルスの変異）が抗タミフル作用が生じるか明らかにした

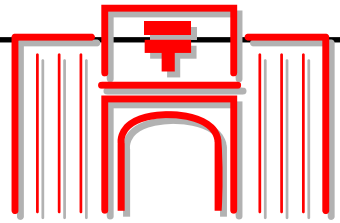

1918 H1N1
Spanish Flu
2003 H5N1
Avian Flu
2009 H1N1A
Swine Flu

# 我々の貢献...

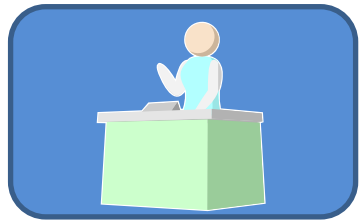
我々が提案したアイデアや開発したもので、
「大規模応用システム」の先行研究・技術と
なったものを中心に...

# 郵便局をCOでモデル化（1977）

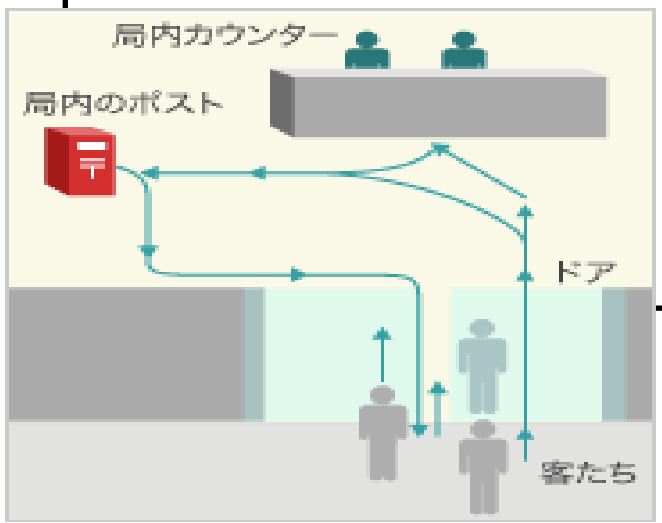
## -- モバイルオブジェクト--



局内のカウンター部

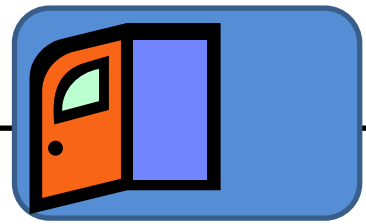


局内のポスト

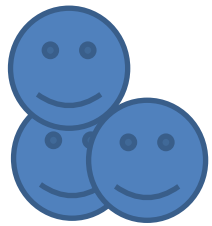



小さな郵便局のモデル。利用者、
局内のポスト、手紙、カウンター、ドアの
いずれも並列オブジェクトで表現できる

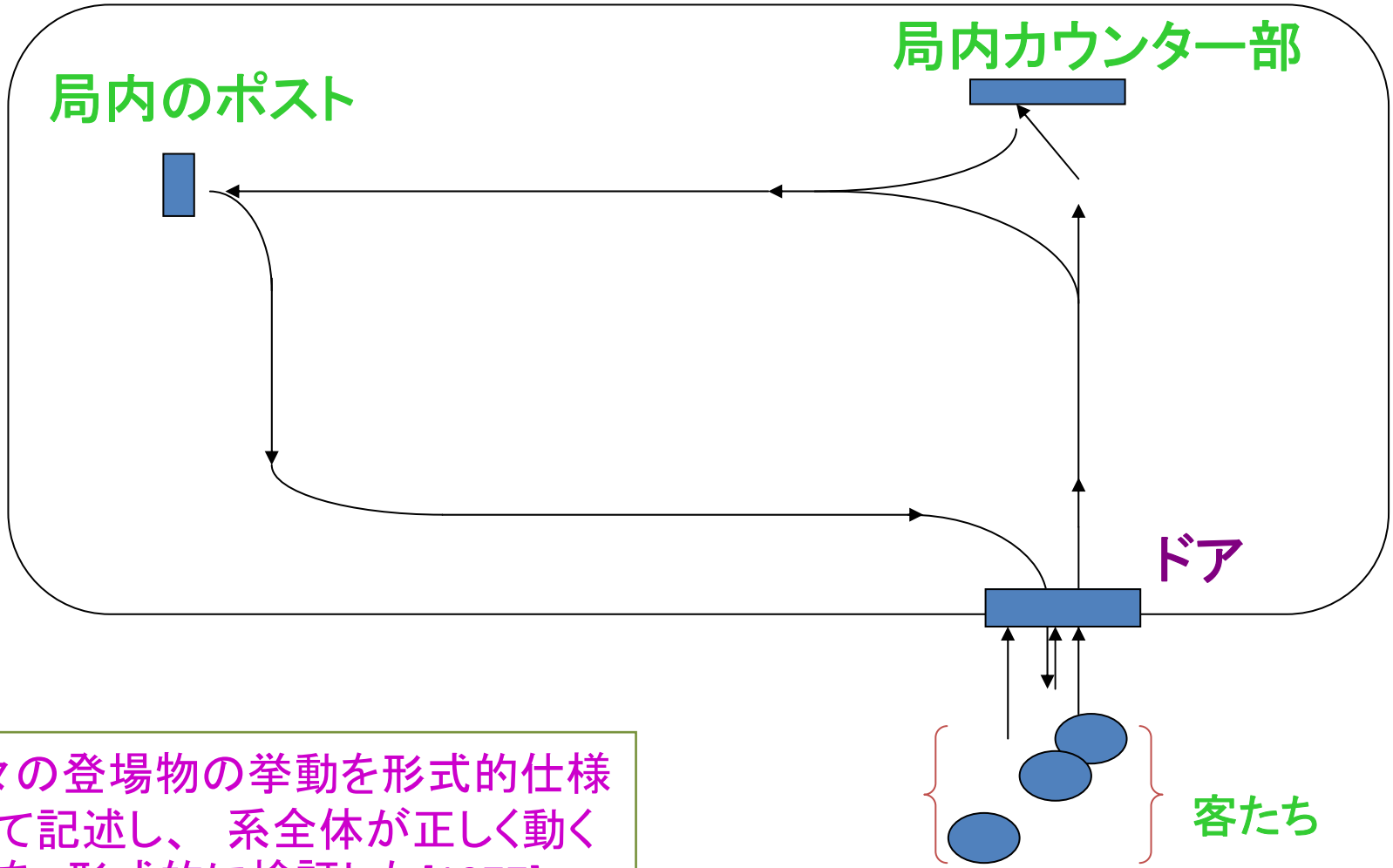
ドア



お客たち



# 郵便局をCOでモデル化・検証(1977)




個々の登場物の挙動を形式的仕様として記述し、 系全体が正しく動くことを、形式的に検証した[1977]

# 小さな郵便局を並列オブジェクトでモデル化する

- 郵便局の建物 ⇔ ドア
  ➔ ドアを表現するCO
- 局員のいるカウンター
  ➔ カウンターを表現するCO
- 郵便ポスト
  ➔ ポストを表現するCO

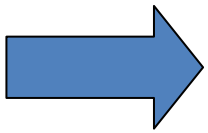
何れも、「状態」を持つ並列オブジェクトでモデル化・表現

- お客たち
  ➔ 並列オブジェクト群で

（注意、メッセージで表現しないこと）

# お客の移動をモデル化する

- ２つのやり方:
  1. 客オブジェクトを
     メッセージに入れてメッセージを送信する
  2. 客オブジェクト自身が自分で移動する



オブジェクト（あるいはそのコード）が
移動・移民(migrate)する!!

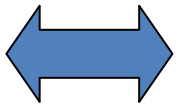
**object as message**

and/or

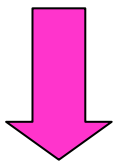
**message as object**

# 環境・アンビアントから教えられる…

- 地元に住んでいない客は
  郵便局内の様子を知らない



客オブジェクトはカウンターオブジェクト
の存在や位置を知らない

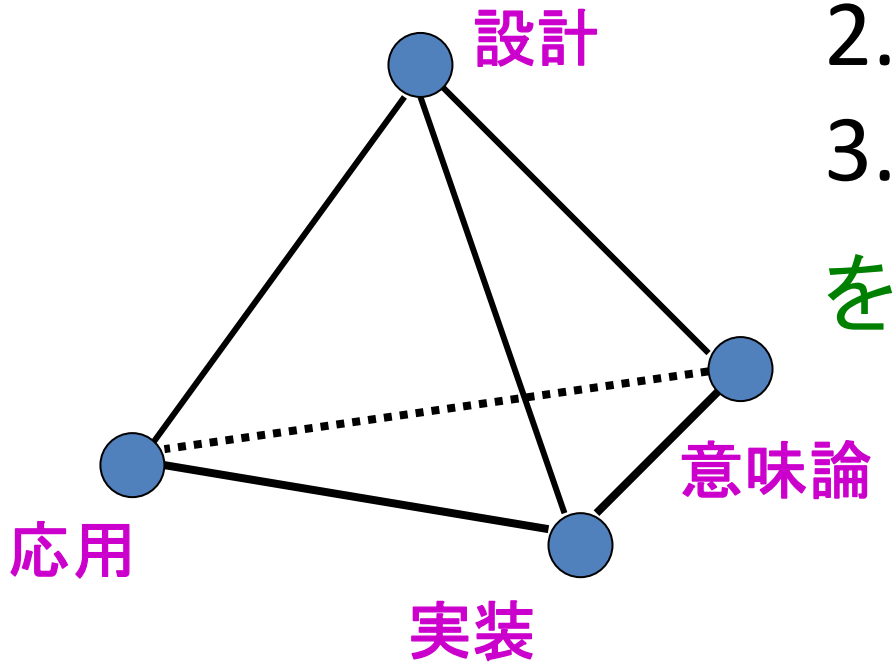


客オブジェクトは、カウンターオブジェクトの位置や
名前を例えばドアオブジェクトから教えられる。

アンビアント

# プログラム言語の正四面体

プログラミング言語を設計したなら、




1.処理系の実装
2.数学的意味論
3.応用プログラム

を、きちんと研究すべし！

# 1984年以来の私のGRの研究

言語設計

- ← *ABCL言語シリーズ* (JP Briot, 柴山) 1984-
- ← Inheritance Anomaly (松岡, JP Briot) 1985-
- ← **自己反映計算** (渡辺) 1987-

意味論

- ← **線形論理を基礎に** (小林) 1991-

超並列コンピュータ上での処理系実装

- ← StackThread**方式** (田浦) 1993-

移動オブジェクトとその実現

- ← JavaGo (関口, 増原) 1999-

応用プログラミング

- ← **N-体問題, 宇宙ステーションの動力学**, CFG**解析（米澤,八杉）**
  … 1987-

# Thanks to 共同研究者たち

柴山悦哉

J-P. Briot

松岡聡

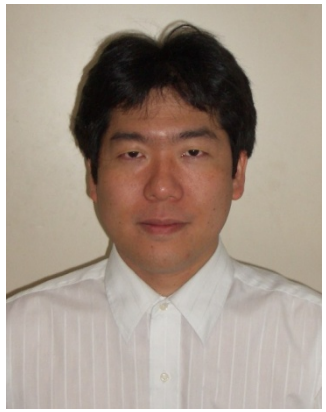
小林直樹

田浦健次朗

増原英彦

渡部卓雄

八杉昌宏



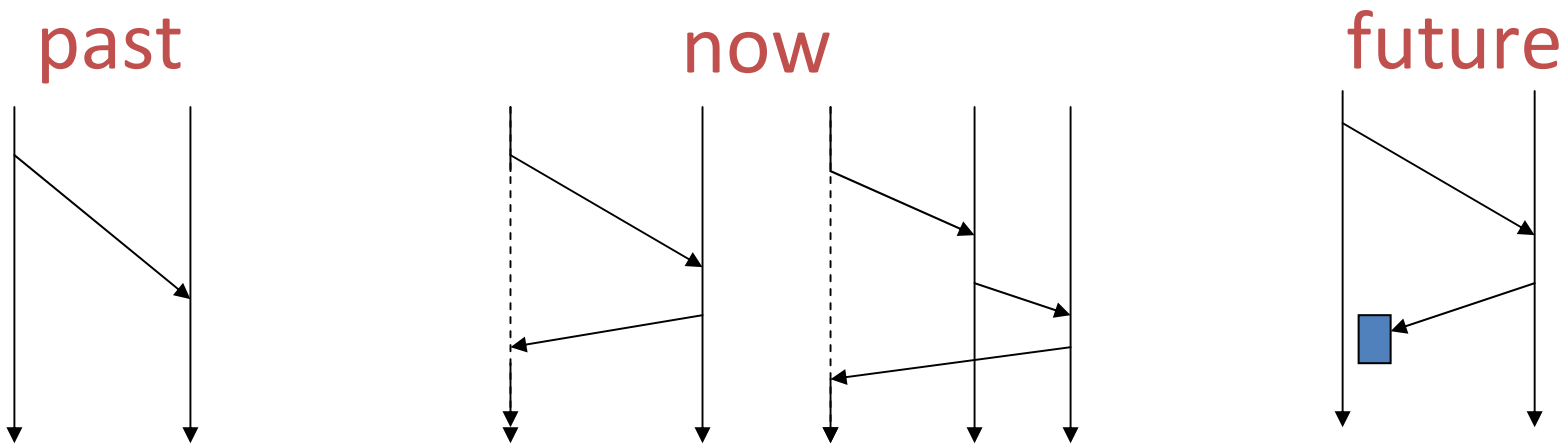
関口龍郎

# 我々の最初の言語 ABCL/1 (1984)

- 使える世界で初の並列オブジェクト言語
- それぞれのオブジェクトは1つのスレッドを持つ
- 任意の時点で, 並列オブジェクトは
  次の3つのモードの 1つにある:
  (1) *休眠*, (2) *活性*, (3) *待機*
- 継承機構を持たない

# ABCL/1言語でのメッセージのやりとり

- メッセージのやりとりは全て非同期式
  - より自然なモデル化とより高度な並列性
- メッセージ送信の 3つのタイプ :
  - 送りっぱなし (past)
  - 送って返事が来るまで待つ (now)
  - 「未来」付き送りっぱなし (future)

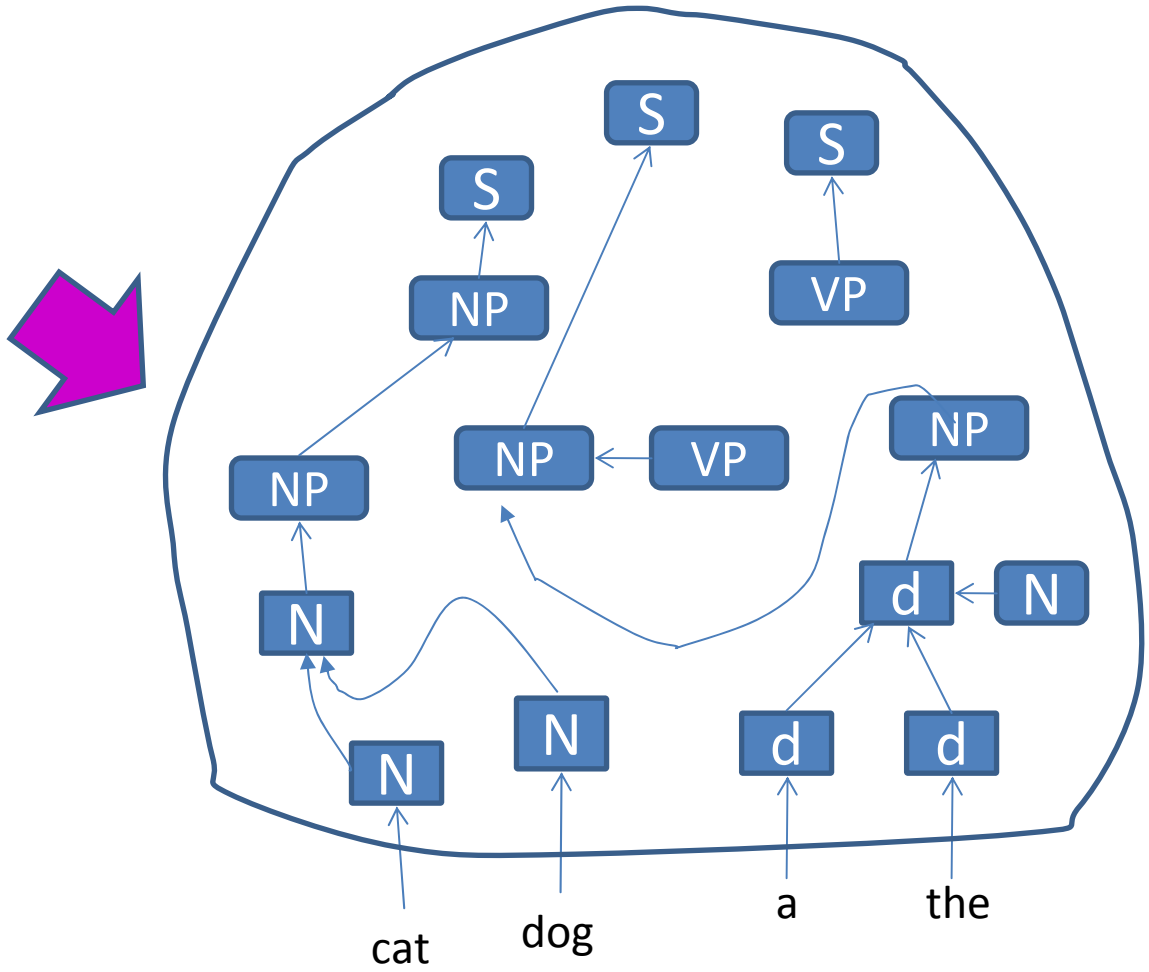


# ABCL/1の
# 処理系実装と応用プログラム

- LISPベースの実装（SUN ワークステーション上）
  - マニュアルとプログラミングガイドをつくり
    OOPSLA'86で配布
- Lispマシン上での より強固な処理系の実装1987.
- 文脈自由文法の並列オブジェクトベースの構文解析器
  - **250以上の非終端記号をもつ英文法（1987）.**
  - ブタペストでの計算言語学会1988に発表、反響あり
    (Computational Linguistic Conference in Budapest, 1988)
  - その後、ヨーロッパから出版された並列構文解析に関する本にヨーロッパ研究者の関連論文が多く収録された。

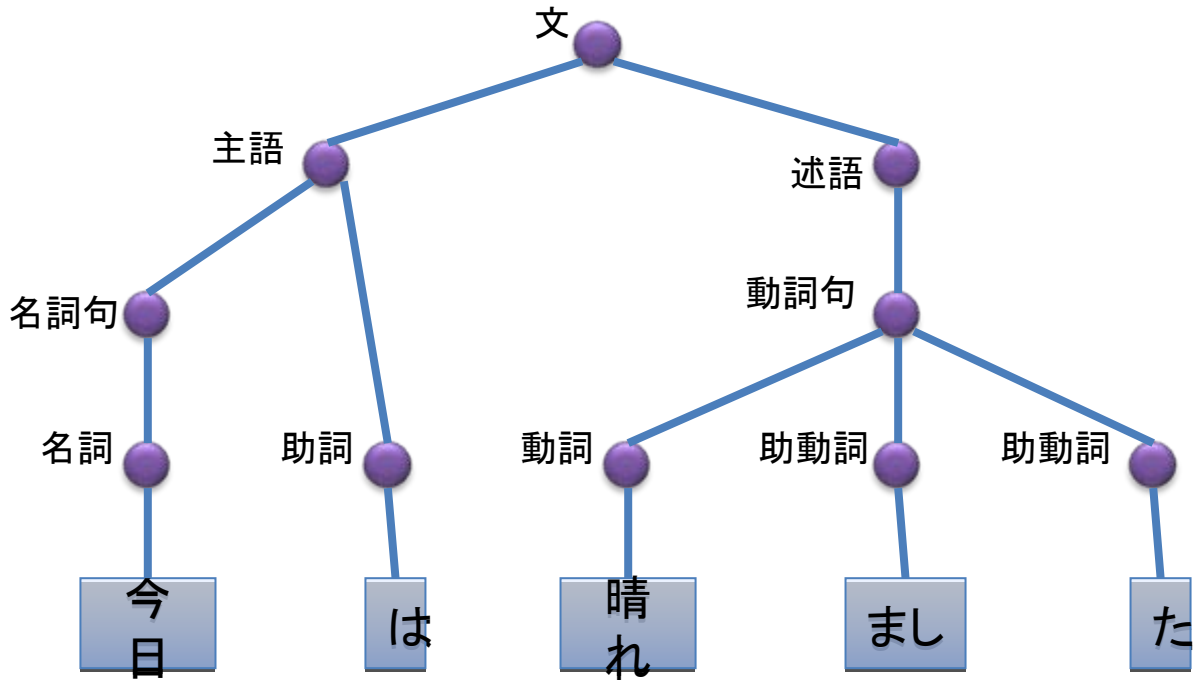
# CFG構文解析器

| | |
|---|---|
| S -> NP | N -> cat |
| S -> VP | N -> dog |
| S -> NP VP | Det -> a |
| NP -> N | Det -> the |
| NP -> Det N | V -> runs |
| VP -> V | V -> saw |
| VP -> V NP | |



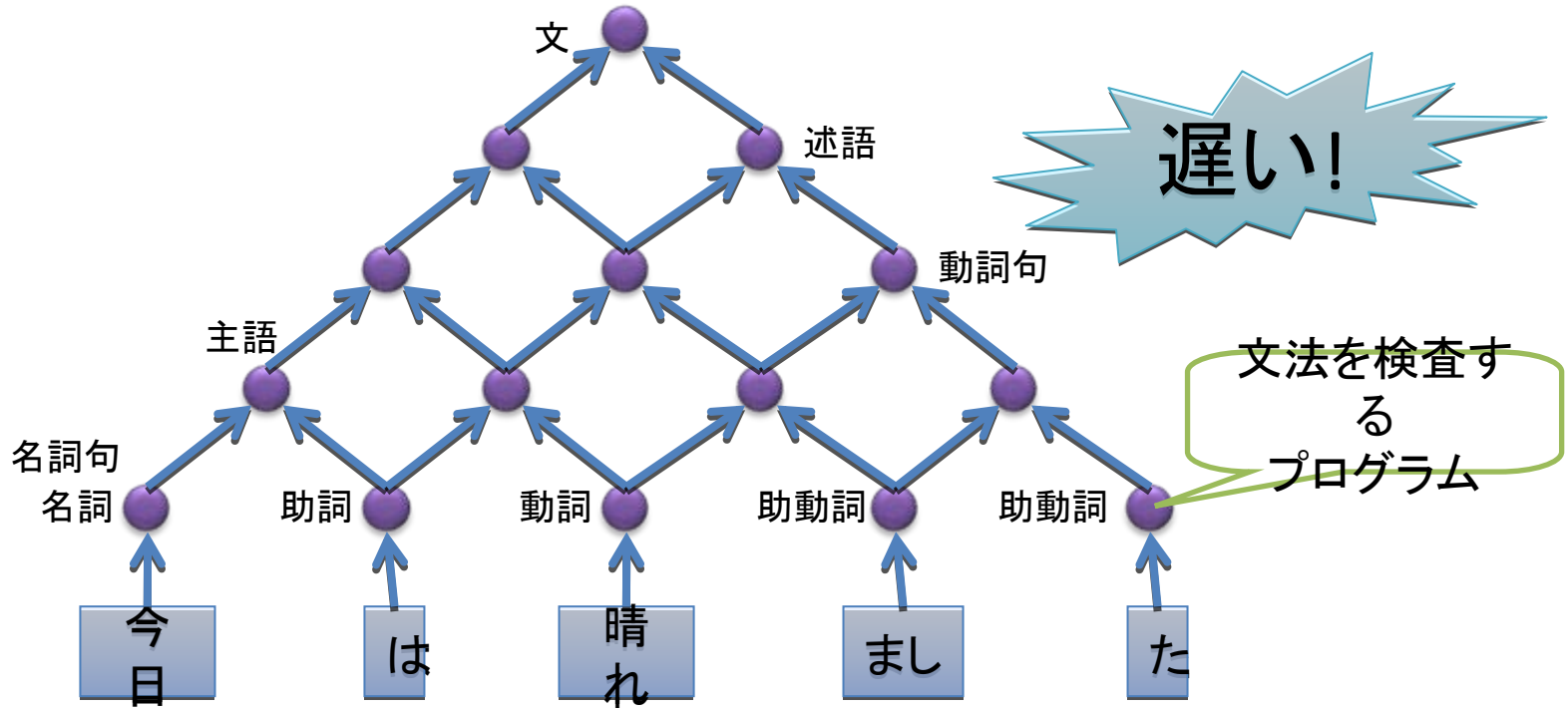
# 文脈自由文法の構文解析問題

- ある文が、ある文法に従った
  正しい文かを判定する問題



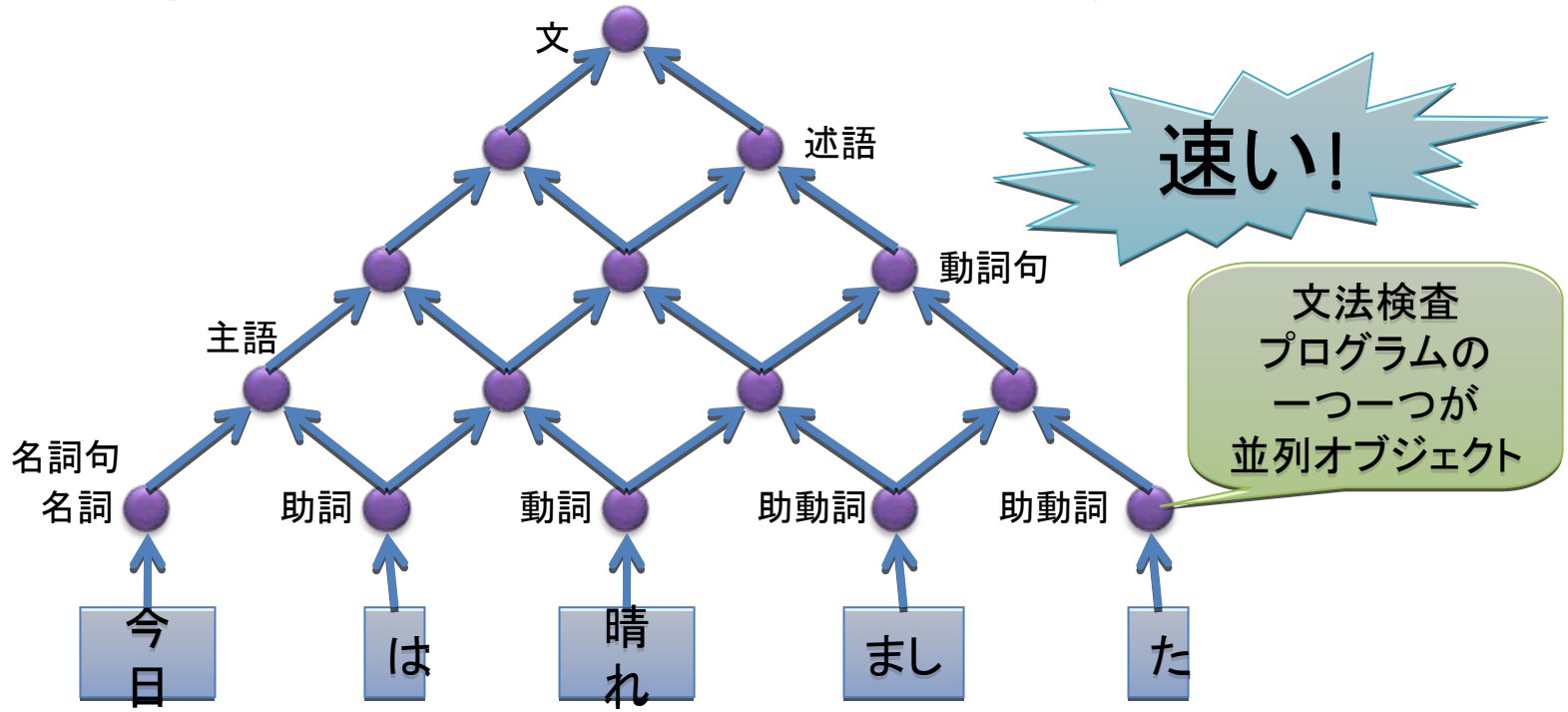
# 並列化をしない
# 構文解析問題の計算方法

- 文法に従っているかを文法規則を一つ一つ順番に検査しなければならず、遅い



# 並列オブジェクトを用いた構文解析問題の計算方法

- 文法を検査するプログラムを「並列オブジェクト」として実装する
  - 文法の検査を同時に処理できるので、速い



# 並列オブジェクトの自己反映計算[1987]

- B. Smith, 3-Lisp(1982)に触発された
- P.Maes & L.Steels, 3-KRS(1984）に触発された
- 渡部卓雄(東工大)と共同で。。。

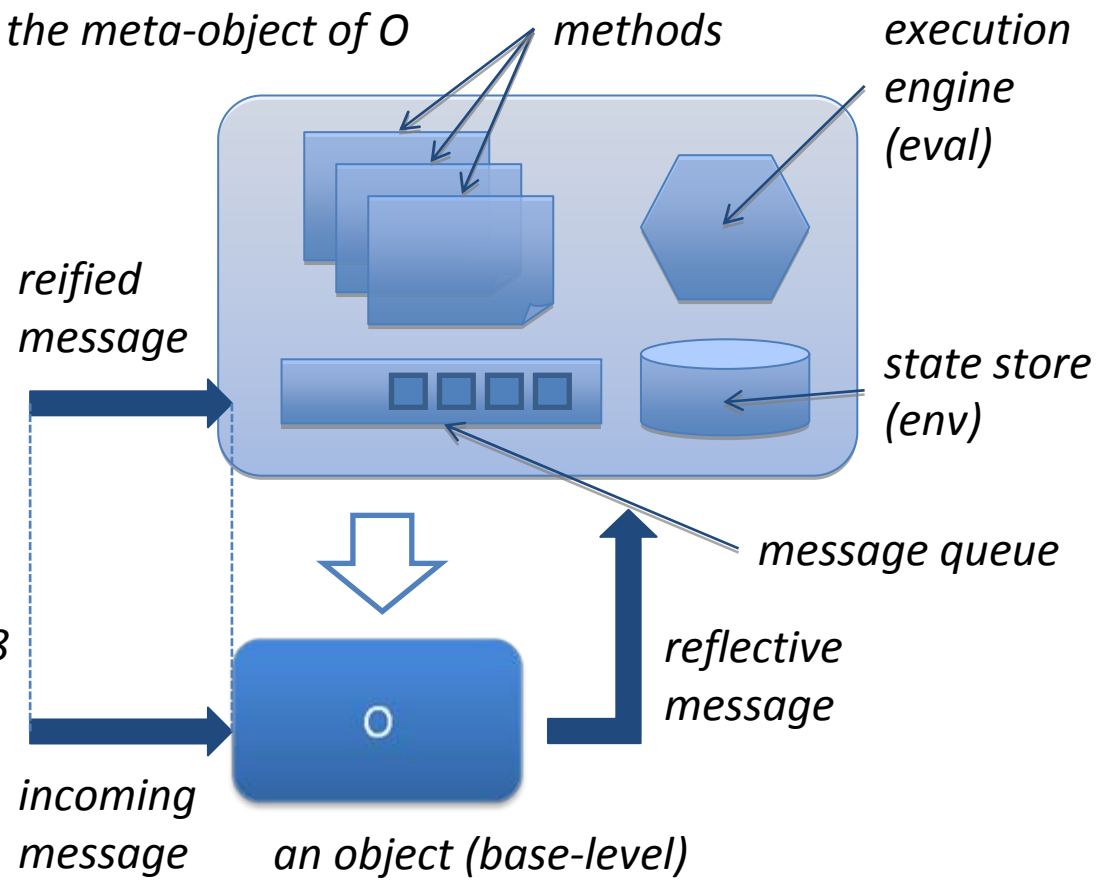
- 並列オブジェクト言語の解釈器(interpreter)を並列オブジェクト言語で記述した
- その解釈器のパーツを実行時に変更・更新することが可能であることを実証した

# ABCL/R言語

並列オブジェクトの解釈器を記述してみる

1つの並列オブジェクトA ｜ー＞1つのメタオブジェクト（Aの実装）

この言語によって、
並列・分散システム
の様々な実装モデル
は設計が記述できる
ことが分かった




*Watanabe & Yonezawa, OOPSLA '88*

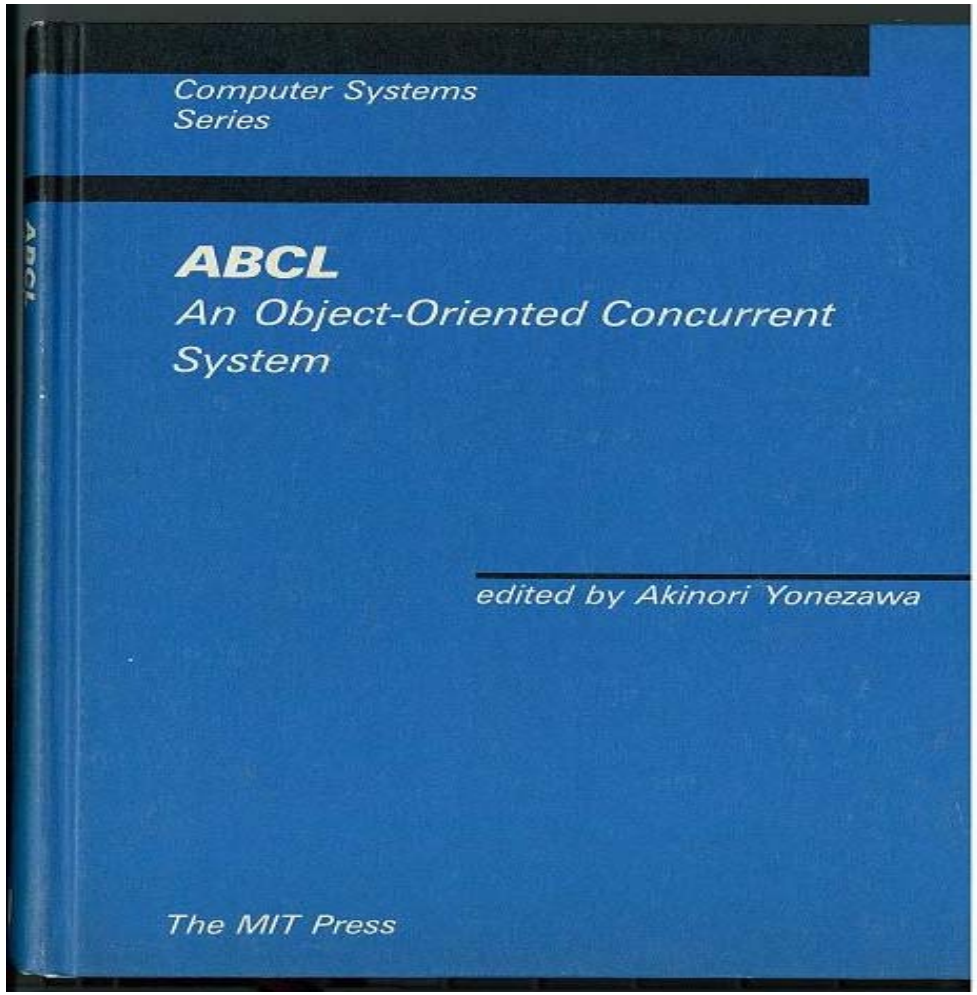
# ABCL/Rを用いた応用

- 動的にメッソドを獲得できる
  - レヴェル間のメッセージのやりとりの例
- 着信・送信するメッセージの監視器の動的着脱
  - メタメタオブジェクトを使って、そのオブジェクトに/から到着/発信されるメッセージの監視するモニターを着脱できる
- メッセージ送受信機能へのタイムワープ機構のモジュラーな導入
  - メタオブジェクトを使っ、分散時間管理システムのアルゴリズムを導入・変更・更新できる

    （例えば、D. Jefferson のTimewarp Algorithm, 1985）

# Book in 1990

- 1989年までの我々の研究論文や言語マニュアルなどを収録した本.



# 線形論理を基礎とする意味論

- 並列オブジェクトに基づくプログラム言語の形式的･数学的意味論が欲しかった！

- R. Milnerのπ-計算を用いて構築するのも1つのやり方ではあったが。。。
  - π計算の帝国に紛れ込むのはちょっと心残り
  - Gentzenスタイルの sequent形式になれていた

⇒

Girardの線形論理を使うことにした！

# 並列オブジェクト指向言語の意味論[1991]

- 小林直樹さんと共同
- 目標:
  - 並列オブジェクト指向言語の数学的基盤を構築する:
    - 言語設計の指針得る(型システムの構築を含む)
    - コンパイラがするコード最適化の正当化
    - プログラム正当性の証明
  - *研究上のprestige*

# 線形論理とは？

- 資源の消費を意識する 論理体系[Girard 87]

  A –o B　　線形含意

  Aを消費することによってBが導出される

  （1度しかAであることを推論に用いない）

  A ⊗ B　　テンソル積

  A と B が同時に成立する

  A & B

  A と B が成立するが, but 同時には成立しない
  (AかBのどちらかを選択しなければならない)

  !A

  Aは何度でも成立する。何度も推論の使える。

# 線形論理のエッセンス

A ： 100円持っている
B ： （100円の）コーラが手に入る
C ： （100円の）チョコレートが手に入る

A –o B　　常に真
A –o C　　常に真
A –o B⊗C　　いつも真ではない
(100円でチョコレートとコーラは同時には買えない）
A –o B&C　　常に真
(100円あれば、
チョコレートかコーラかどちらかは必ず買える)

# 並列オブジェクトを線形論理式として表現する

- *m -o A*
  - メッセージ mを*受理/消費して*、その後 Aのように振舞うオブジェクト
- *m ⊗A*
  - メッセージmを*送信し*, その後 Aのように振舞うオブジェクト
- 論理推論を計算とみなす(deduction as computation)

(c.f. logic programming)

$$\underbrace{(m \otimes A)}_{\text{送信者}} \otimes \underbrace{(m \text{ -o } B)}_{\text{受信者}} \text{ -o } A \otimes B$$

# 並列オブジェクトを線形論理の論理式として表現する

$$!\forall n,inc,read.$$
$$(counter\ (n, inc, read)\ \multimap$$
$$\forall reply.(inc\ (reply)\ \multimap$$
$$(counter\ (n+1, inc, read) \otimes reply()))$$
$$\&$$
$$\forall reply.(read\ (reply)\ \multimap$$
$$(counter\ (n, inc, read) \otimes reply(n))))$$

# 超並列マシン上での言語処理系の実装[1992]

–田浦健次朗さんと共同

# F-社製AP1000超並列マシン（512ノード）上、ABCL言語の実装技術の研究開発(1992-)

- 超並列マシン上で並列オブジェクト指向言語 ABCL/f　言語の処理系の一連の開発
- ハイパフォーマンス計算を狙って開発開始

(超)分散メモリ型並列マシンをターゲットとしたハイパフォーマンス計算用の並列言語処理系としては、初期の数少ない開発

[PPoPP'93, PLDI'97,PPOPP'99]

# 我々が辿りついた方式は。。。

- *並列オブジェクト = データ + 1つの実行スレッド*
  - しかしこの考えでは旨く行かなかった、なぜなら スレッドがとてつもなく資源を使う

通常のライブラリとして提供されるスレッド
ex. pthread

*。。。いろいろ試みた。。。*

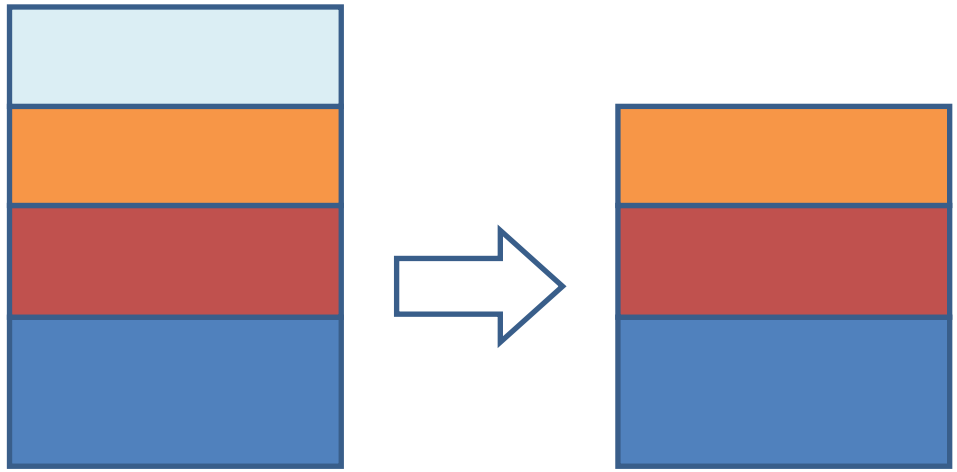
- 試みN: “スタックスレッド”方式というものを考案
  - *1つのスタックだけを使って全てのスレッドを投機的に実行する*

# “StackThreads方式”

- 唯一のスタックで全てのスレッドを実行
- では、スレッド切り替えをどうするのか?
  - 単純!、スタック内ポインタの操作だけで行い、かつスタックフレームの削除はスタックトップから行う
- 大変低コスト、かつ高速のスレッドが得られた

=>

この結果、莫大な数の細粒度スレッドが利用可能となった。



# StackThreadsで分かったこと

- 並列オブジェクトは細粒度スレッドプログラミングに非常に適した枠組みである！！

See http://www.yl.is.s.u-tokyo.ac.jp/sthreads/
これは細粒度マルチスレッドライブラリで
GCC/G++でも使える。今でもダウンロードで使用可能

  - GNU C コンパイラをバックエンドとし利用
  - parallel C/C++ で使えるようにした

# 並列オブジェクトの我々の応用例

- Barnes-Hut アルゴリズムを用いた
  多体問題シミュレーション
- 宇宙ステーションの制御
- 並列オブジェクトベースの文脈自由文法構文解析器

# 並列オブジェクトによるN体問題simulation

- *並列オブジェクトに:*

  - 星(質量)　や　星群の重心

  *を表現させる！*

- *各並列オブジェクトがその状態で:*

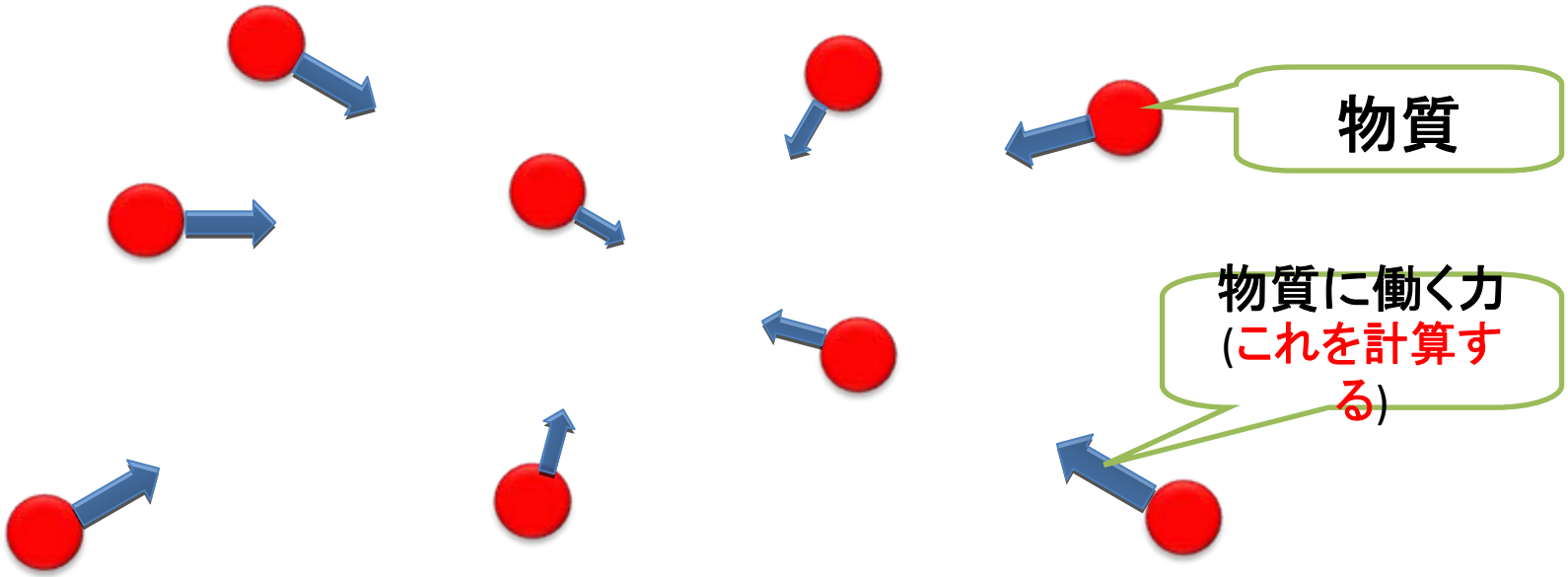
  - xyz-位置, 速度, 質量

- Barnes-Hut法を使用

- 1995年に, StackThreadsベースのコンパイラを用いて512 SPARC nodesのAP1000で実行

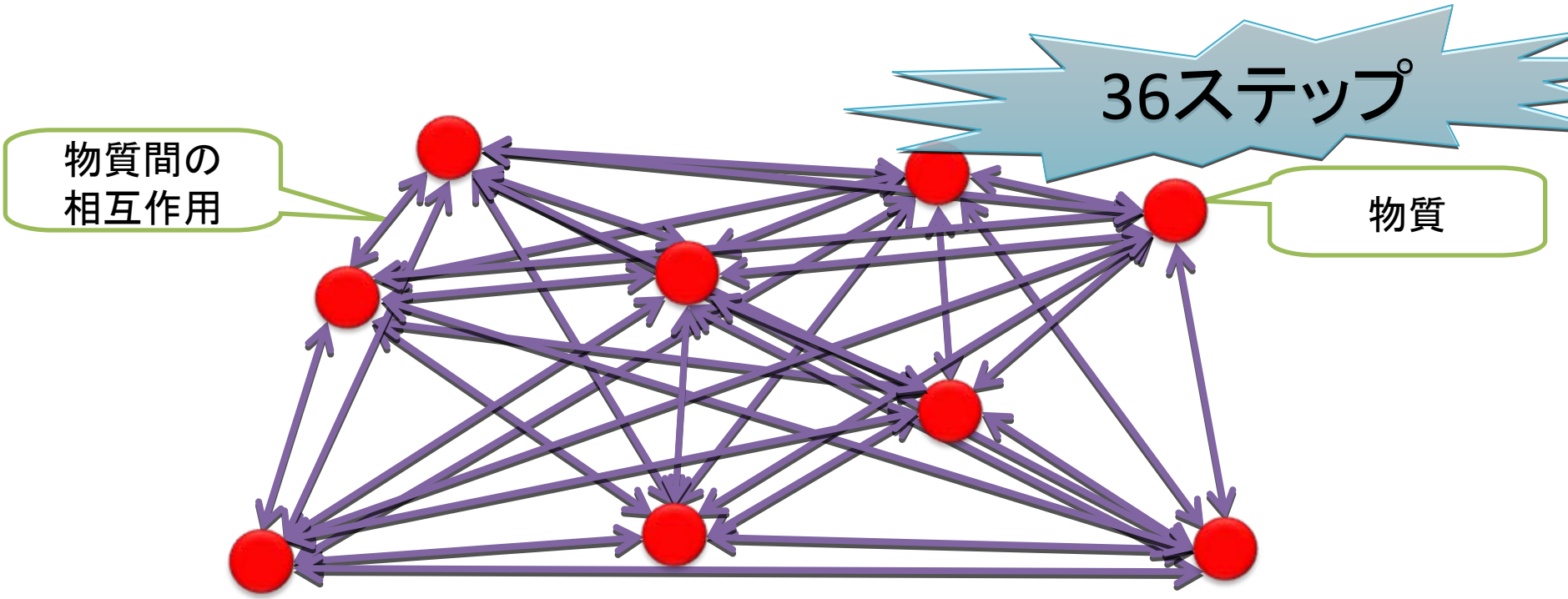
# 多体問題とは

- たくさんの物質 (質点) 間の万有引力を計算しその運動をシミュレートする問題
  - 銀河の形成など宇宙の構造形成の研究等で用いられる



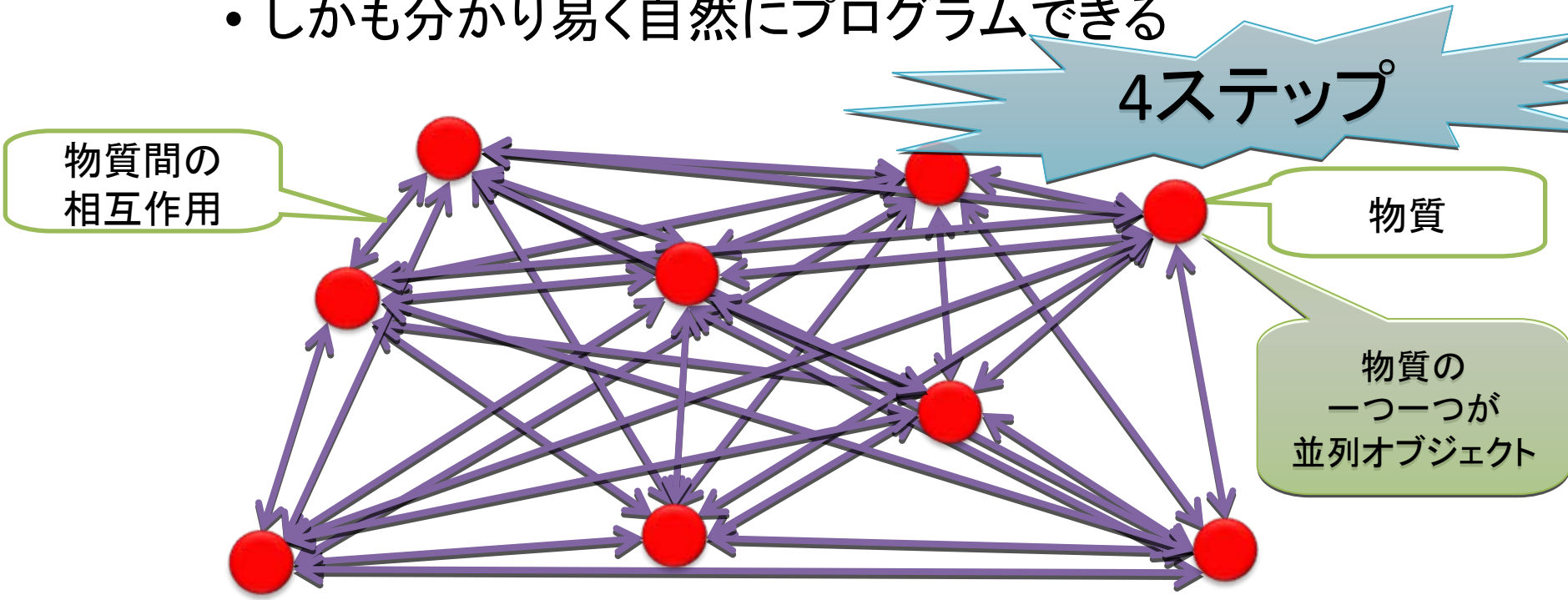
# 並列化しない
# 多体問題の計算方法

- 全ての物質間の相互作用を
  一つ一つ順番に計算しなければならず、遅い

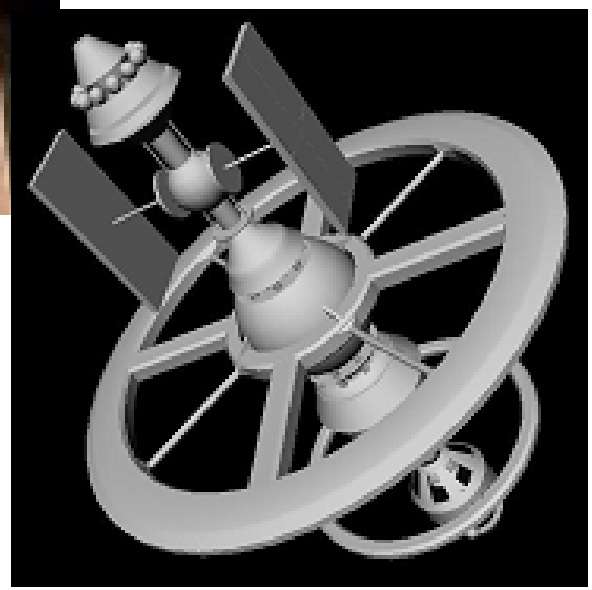


# 並列オブジェクトを用いた多体問題の並列計算方法

- 各物質を「並列オブジェクト」として計算する
  - 物質間の相互作用を同時に計算できるため、速い
    - しかも分かり易く自然にプログラムできる



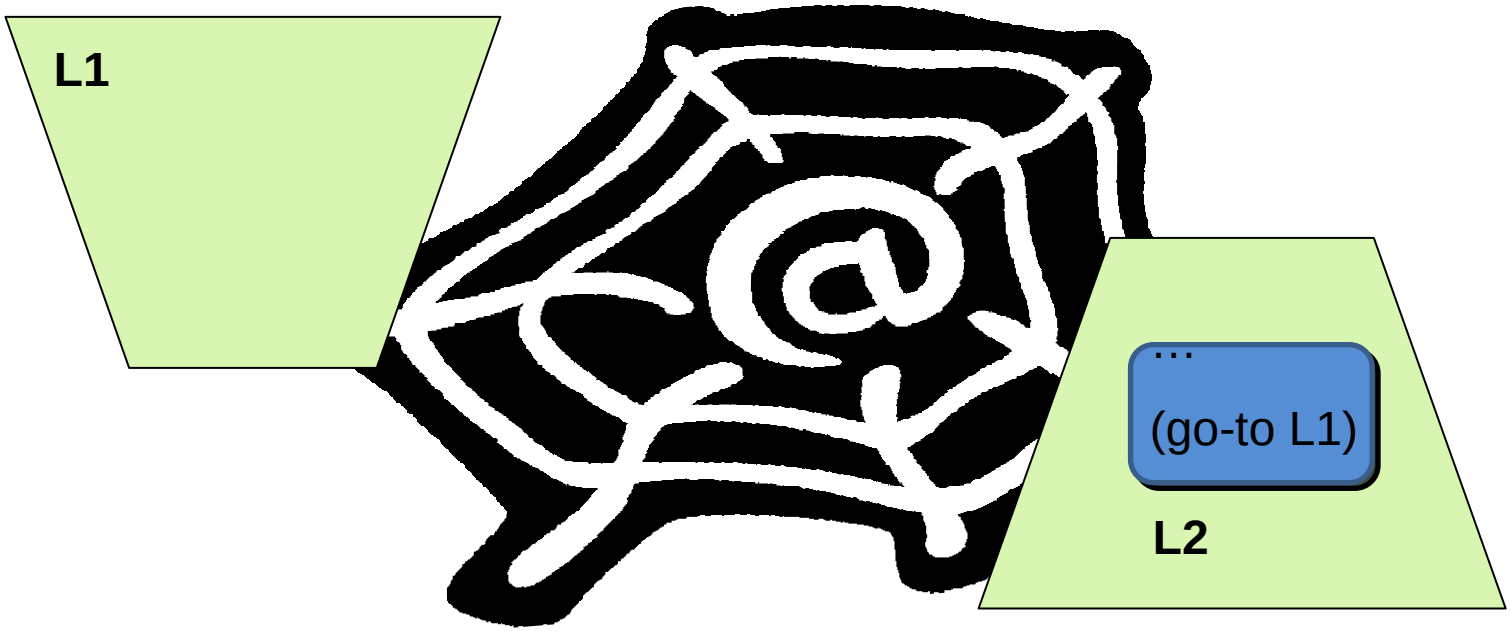
# 宇宙ステーションの動力学と制御



- 剛体と結合を並列オブジェクトで表現する.
- 並列オブジェクト群がトルクや力を計算して宇宙ステーションの動きを計算あるいは安定させる力のかけ方を計算する.

# 自律的に移動するオブジェクト

# 自立的に移動可能な並列オブジェクトを用いるプログラミング– JavaGo言語[1999]




- ネットワーク上を自由の自立的に
  移動可能な並列オブジェクトをサポートする
  JavaGo 言語とその実装 (1999)

# JavaGoX: オブジェクトの自律的な移動を簡単にプログラムできる言語[2000]

- オブジェクトの移動をサポートするJava機能
  - 動的なクラス定義のローディング
  - オブジェクトの逐次化(serialization)
- JavaGoX は、現在実行されているオブジェクトを効率よく移民させることができる。
  - 実行スタックをセーブしたり回復したりするコードに直前・直後に挿入する。
  - JVMバイトコードの変換手法をもちいた実装

Sakamoto, Sekiguchi, Yonezawa:

*Bytecode Transformation for Portable Thread Migration in Java, in ASA/MA2000（最優秀論文賞）*

# 超並列、many-core時代に入って

-- スパコンは「超並列」型が全盛に

-- ワークステーションクラスタ利用者が
超並列マシンへ移行。。

# マルチコアマシンの性能を極限まで引き出すには

- 2, 4, 或いは 8 way のマルチコアをノードとするマシンが入手可能になっている
- 性能を極限まで引き出すには、
  1. 超細粒度の軽量スレッドが手軽に使えて、
  2. 共有メモリによらない、通信方式が必要になる
  3. それも僅少なコストで!!

それが今、可能になりつつある!!

# 並列言語が戻ってきた

- “並列言語” は長く, 隙間研究（nichie)であった!
  しかし今は,
  - Scala言語、Erlang言語、。。。
  - X10言語、。。。

  などをはじめとする多くの言語が、
  並列プラットフォームの上でその実装が進み、
  実用化が進んでいる。。。

# 超軽量並列が再び脚光を

- 軽量スレッドは安く使えるようになった。

- “超軽量並列が” 古くからのイアデアであるが、
  プログラミング言語コミュニティが貢献できる
  強力なテクノロジーであるので
  このアイデアは
  今後広く使われると予想される！！

# ErlangやRevactorになぜ並列オブジェクトが

- Erlang:　WEBアプリや
  分散、フォルトトレラント計算用に人気
- Revactor: 並列オブジェクト・アクターモデルのRuby用
  の実装で、WEBアプリで人気
- 両者が使っているのが:
  - 非同期方式のメッセージ送信によるコミュニケーションで、
  - 超軽量スレッドによる、mailboxを介してのsend/receive操作
- なぜか？
  - 「ロック・解除」操作が不要になる！！！

    => 容易で安全は細粒度並列プログラミング!!!

# X10: The Concurrent Programming Language for Multi-Core & Petascale Computing

- IBMがBlueGenesを含む将来のスパコンに乗せようとしている言語
- DARPAのHigh Productivity Computer Systemプログラムの傘下で、Productive Easy-to-use Reliable Computer Systemsの一環として設計・実装 (HPC Challengeで受賞)

- X10は、Javaプログラマにとっても優しい、
  型安全、並列・分散オブジェクト指向言語
- 特徴:
  非同期の通信と計算 (futureもある)
  分割された大域アドレス空間の導入

# おわりに

# 並列オブジェクトのポイント

1. メッセージ(イベント)駆動型処理(message-driven)
2. 非同期メッセージ送信(asynchronous comm.)

3. 情報の交換・共有手段はメッセージのやりとりのみ
4. 制御の流れとデータ・情報の流れが一致してる

5. 多数の並列オブジェクトを用いて(超)高度並列性

6. 安全な軽量プロセス(light-weight process)
   - lock, unlock操作が不要

7. 自然なモデリング
   自然な分割の単位　ー　機能(データとプログラム)

# 並列オブジェクトの時代？

- 「オブジェクト指向」

  Java/C++　　90年代半ばに確立

  そこでやっと「OO」が完全に受け入れられた！

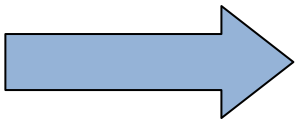
- 「並列オブジェクト」
  - 超軽量スレッドが手軽に使えるようになった
  - メニコア化が今後進むと

    「CO」もどんどん受け入れられるのでは！？

# もっと前進したい…

“並列オブジェクト” が享受するものは:

- 自然かつ強力はモデル化能力,
- 容易で安全な 並列性とスレッドの活用・管理
- 超並列軽量スレッドの実装技術、
  （例えば、*StackThreads*） が実現されてきた
- マルチコアアーキテクチャが実現され、どんどん使われる.



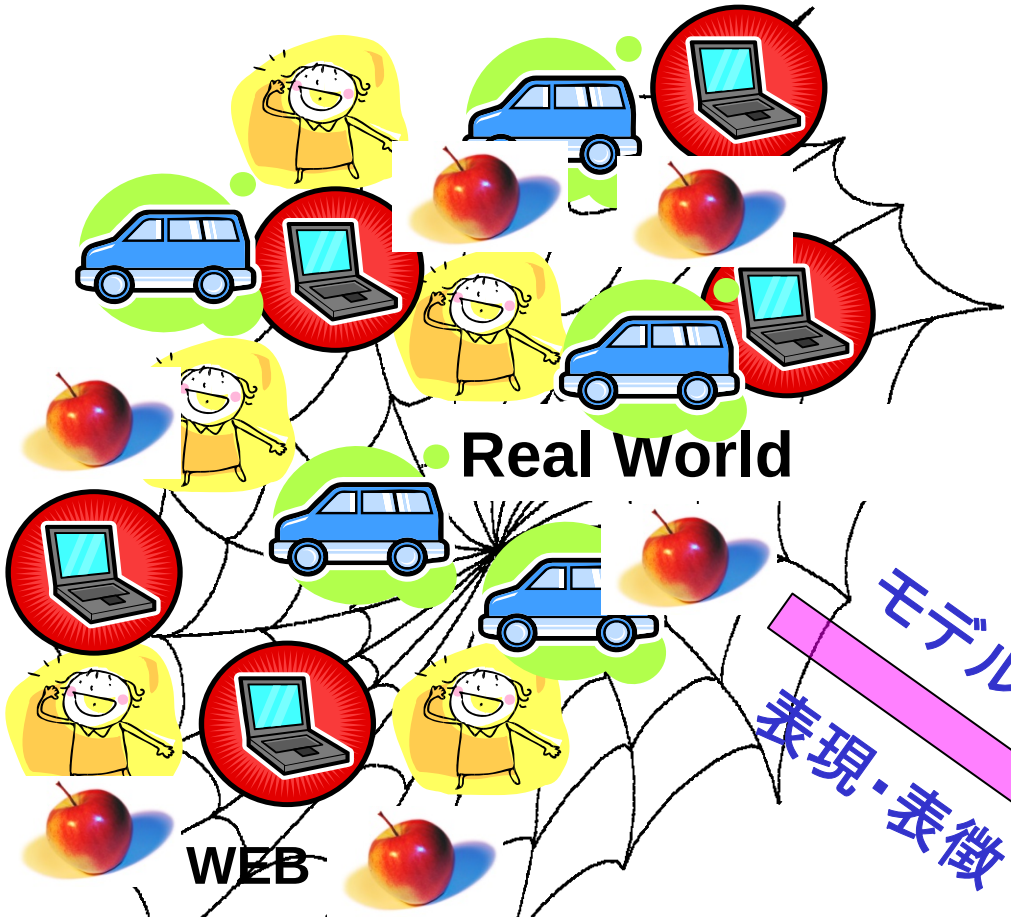
もっと精密でもっと強力な

物理世界、生態系、生物体、社会構造、組織構造

などの実世界や仮想世界の

モデル化・シミュレーション・プログラミングが可能になる

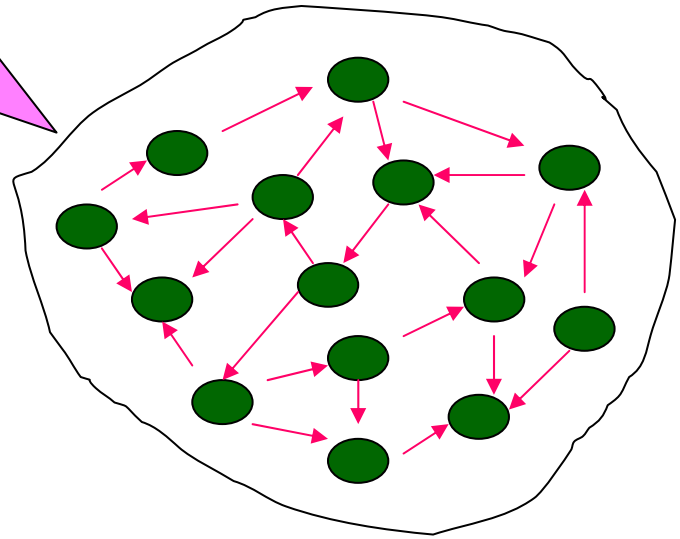
# 並列オブジェクトによるモデル化とシミュレーション




事物,人々,
コンピュータ群 &
それらの相互作用

モデル化
表現・表徴

並列オブジェクト群 &
メッセージのやりとり



ご清聴に感謝