Tools for Large-Scale Spatial Simulation Design and Analysis

Johannes Gehrke

(Joint work with Tuan Cao, Al Demers, Nitin Gupta,
Christoph Koch, Ben Sowell, Marcos Vaz Salles, Walker White, Tao Zou)

Big Red Data Group

Department of Computer Science, Cornell University

http://www.cs.cornell.edu/bigreddata/games

Cornell University

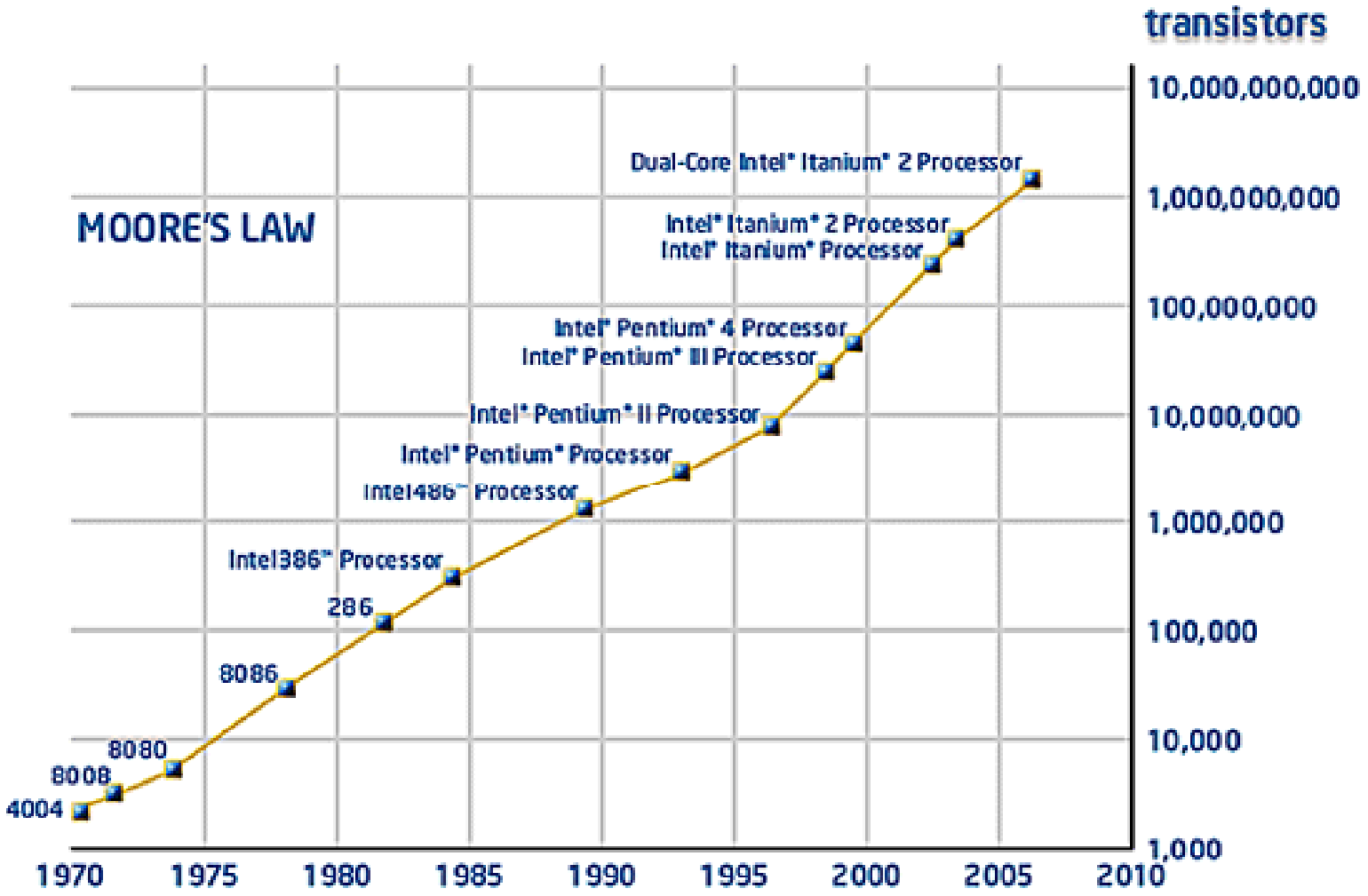
An Abundance of Data

- Supermarket scanners
- Credit card transactions
- Call center records
- ATM machines
- Web server logs
- Customer web site trails
- Podcasts
- Blogs
- Closed caption
- Scientific experiments
- Sensors
- Cameras
- Interactions in social networks
- Facebook, Myspace
- Twitter
- Speech-to-text translation
- Email

•Print, film, optical, and magnetic storage: 5 Exabytes (EB) of new information in 2002, doubled in the last three years [How much Information 2003, UC Berkeley]

Cornell University

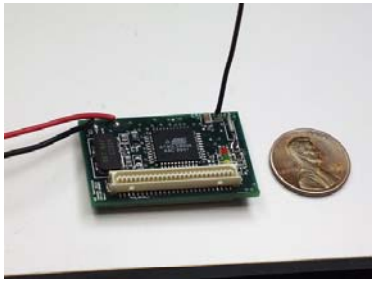
Driving Factors: A LARGE Hardware Revolution



[Intel Corporation]

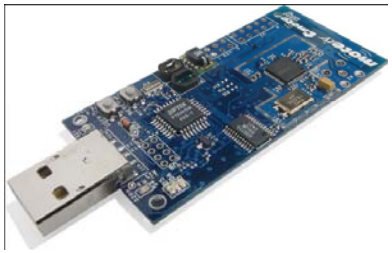
Cornell University

A small Hardware Revolution

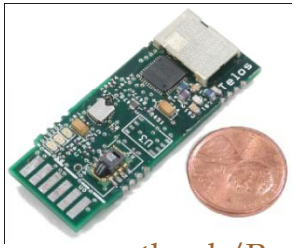


http://lecs.cs.ucla.edu/Resources/testbed/testbed-overview.html

http://www.snm.ethz.ch/Projects/MicaZ



http://www.snm.ethz.ch/Projects/TmoteSky



http://www.snm.ethz.ch/Projects/Telos

http://www.snm.ethz.ch/Projects/Mica2Dot

- Moore's Law
  - In 1965, Intel Corp. cofounder Gordon Moore predicted that the density of transistors in an integrated circuit would double every year.
  - Later changed to reflect 18 months progress.

Driving Factors: Connectivity and Bandwidth

- Metcalf's law (network usefulness increases squared with the number of users)

- Gilder's law (bandwidth doubles every 6 months)

Definition

Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.

Example pattern (Census Bureau Data):
If (relationship = husband), then (gender = male). 99.6%

Why? Three Examples

- Sensor networks
- BIG Science Data
- Photos and videos

Cornell University

Talk Outline

Data

- Sensors
- Science
- Images

Techniques

- Declarative processing

Cornell University

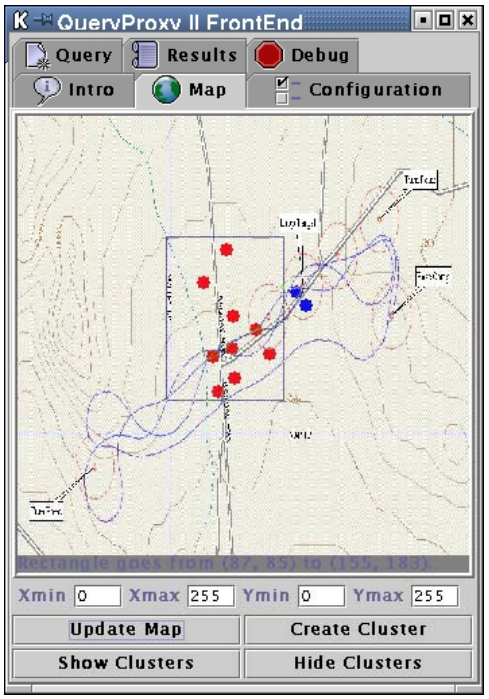
Flexible Decision Support (1999)

Traditional

Procedural addressing of individual sensor nodes; user specifies how task executes, data is processed centrally.

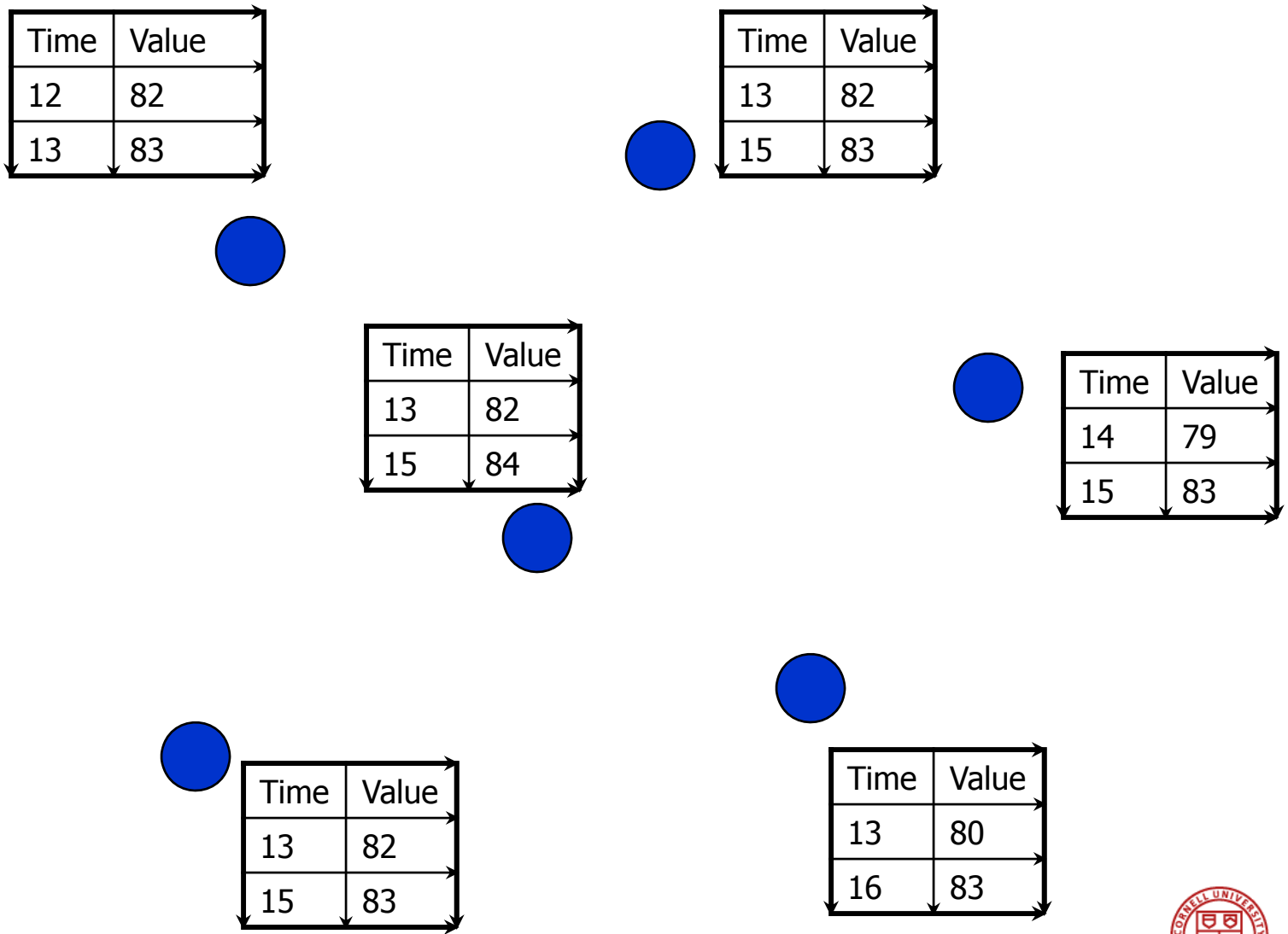
Today

Complex declarative querying and tasking. User isolated from "how the network works", in-network distributed processing.



http://www.cs.cornell.edu/bigreddata/cougar/

Querying: Model



Example Queries

Snapshot queries: In which area is the concentration of chemical X higher than the average concentration?

```
SELECT AVG(R.sensor.concentration)
FROM Relation R
GROUP BY R.area
HAVING AVG(R.sensor.concentration) >
                    (SELECT AVG(R.sensor.concentration)
                     FROM Relation R
                     GROUP BY R.area)
```

Long-running queries: Notify me over the next hour whenever the concentration of chemical X in an area is higher than my security threshold.

```
SELECT R.sensor.area, AVG(R.sensor.concentration)
FROM Relation R
WHERE R.sensor.loc in rectangle
GROUP BY R.sensor.area
DURATION (now,now+3600)
```

Archival queries

Cornell University

Goals

- Declarative, high-level tasking
- User is shielded from network characteristics
  - Changes in network conditions
  - Changes in power availability
  - Node movement
- System optimizes resources
  - High-level optimization of multiple queries
  - Trade accuracy versus resource usage versus timeliness of query answer

Challenges

Technical:

- Scale of the system
- Constraints
  - Power, communication, computation
- Constant change, uncertainty from sensor measurements
- Distribution and decentralization

http://www.fatvat.co.uk/2010/07/stop-traffic.html

Application:

- Environmental monitoring
- Health Care
- Care for the elderly

Talk Outline

Data

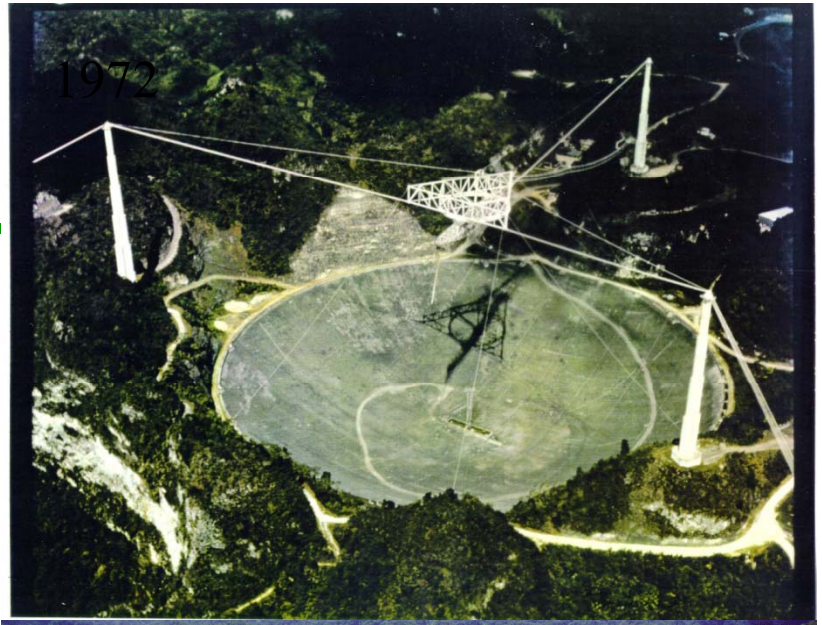
- Sensors
- Science
- Images

Techniques

- Declarative processing

Cornell University


http://www.geoeye.com/CorpSite/gallery/detail.aspx?iid=275&gid=51

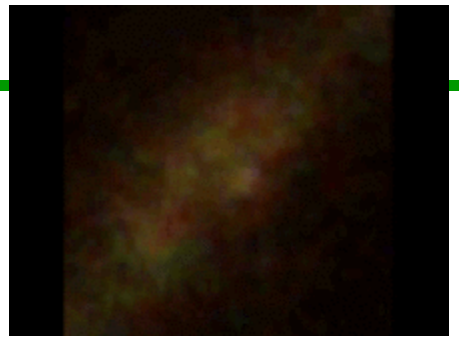


http://www.naic.edu/

Cornell University


http://www.naic.edu/

Pulsar Surveys (2003)



http://en.wikipedia.org/wiki/Pulsar

- Pulsars are rotating stars
- Of interest are
  - Millisecond pulsars
  - Compact binaries
- Example:
  - Hulse-Taylor binary
  - Used to infer gravitational waves in support of Einstein's General Theory of Relativity
  - Nobel price in physics in 1993

Pulsar Surveys (Contd.)

- Part of the ALFA surveys
  - ~ 100 MB/s to disk
  - ~ 1 PB for entire survey (3-5 yr @ 6-10% duty cycle)
- Requires coarsely parallel processing of raw data in discrete, local data chunks
  - processing time ~ 50-200x data acquisition time on single processor (Intel 2.5 GHz 512k cached with 1GB ram)
  - Distributed processing (Cornell + 5 sites)
- Requires meta-analysis of data products of the initial analysis
  - Database and data mining research problems

Challenges

- Data
  - 14 TB every 2 weeks
  - Shipped on USB-2 disk drives
  - Need to archive raw data
  - Need to make data products to the astronomy research community
- Processing
  - Extremely processor intensive
    - Currently just exhaustive search over a large parameter space (periodicity, dispersion, time)
  - Find new pulsars --- and other *interesting* phenomena
- More information: http://arecibo.tc.cornell.edu/hiarchive/

Talk Outline

Data

- Sensors
- Science
- Images

Techniques

- Declarative processing

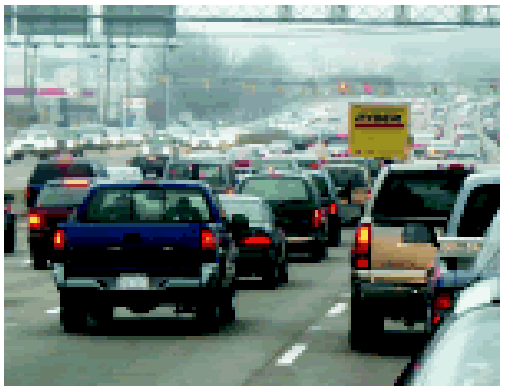
Cornell University

Image Collections (2010)

flickr™

www.flickr.com

www.facebook.com



Source: EPA at http://www.epa.gov/

Cornell University

The Need for Large-Scale Image Processing

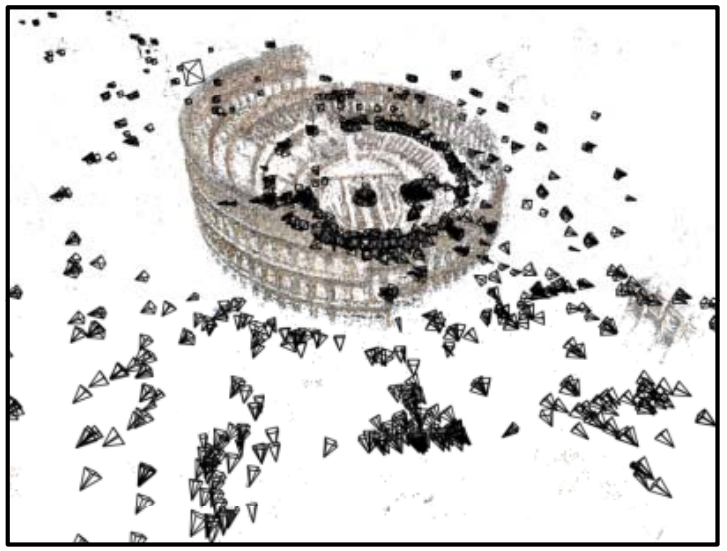
Photos:

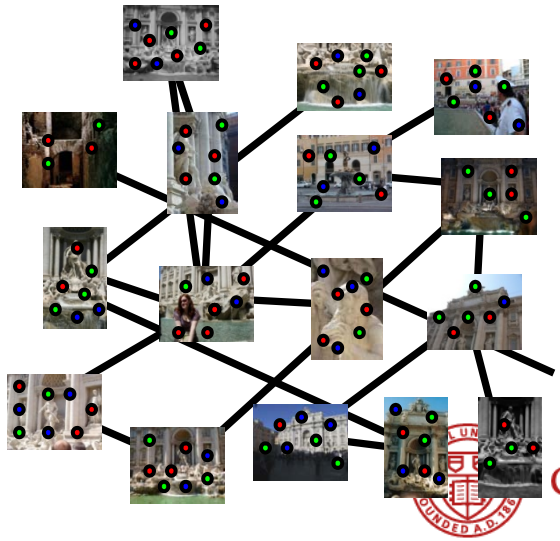
- **5 billion** – Photos hosted by Flickr
- **3000+** – Photos uploaded per minute to Flickr.
- **130 million** – At the above rate, the number of photos uploaded per month
- **3+ billion** – Photos uploaded per month to Facebook.

Video:

- **2 billion** – The number of videos watched per day on YouTube.
- **35** – Hours of video uploaded to YouTube every minute.
- **186** – The number of online videos the average Internet user watches in a month (USA).
- **2+ billion** – The number of videos watched per month on Facebook.
- **20 million** – Videos uploaded to Facebook per month.

The Power of a Data-Rich Environment





Pictures courtesy of Noah Snavely
http://www.cs.cornell.edu/~snavely/

Cornell University

Statue of Liberty

Picture courtesy of Noah Snavely
http://www.cs.cornell.edu/~snavely/

Talk Outline

Data

- Sensors
- Science
- Images

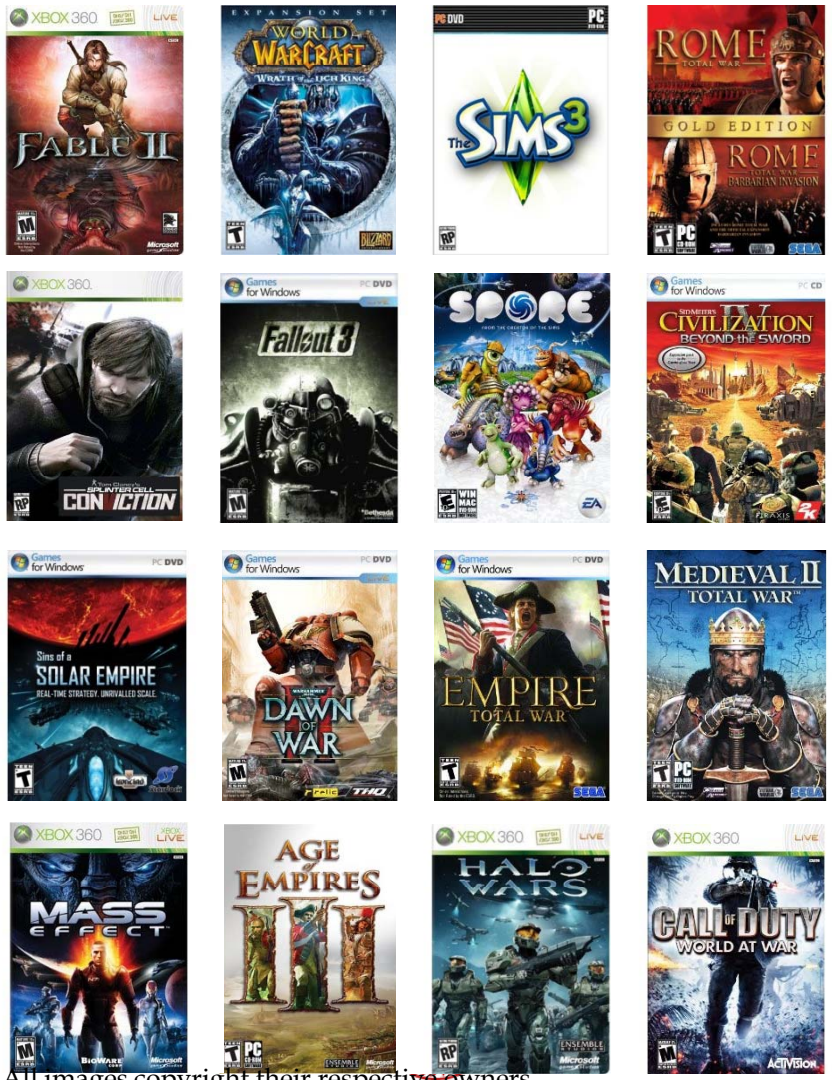
Techniques

- Declarative processing

Cornell University

Video Games

- Virtual environments
- High degree of interactivity



Simulation Games

- What are simulation games?

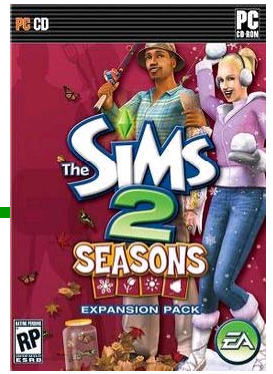
The Sims 3 © Electronic Arts

Warcraft © Blizzard Entertainment

- "Doll House" games
  - NPCs have needs and desires.
  - Objects can satisfy needs and desires.
  - Player control via object placement.
- RTS games
  - Troops move and fight in real time.
  - Player multitasks between large number of units.
  - Player control via a limited number of commands.

Cornell University

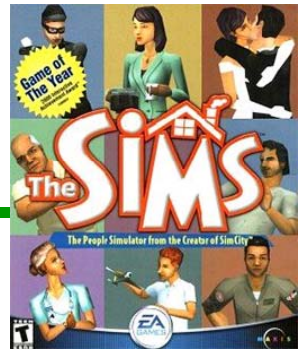
Simulation Game Design: NPCs



- Non-Player Charaters (NPCs): Characters not directly controlled by the player.
  - Main actors in the game
    - *Doll House Games*: Enticing a hungry NPC with some food
  - Enemies controlled by the computer
  - Allies indirectly controlled by the player
    - RTS Games: Issuing commands to military units

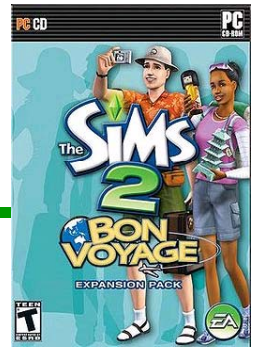
- Simulation games: All characters are NPCs
  - All character actions simulated by computer
  - Player controls everything *indirectly*

Data-Driven Game Design



- Game design brings together many disciplines
  - Art, design, storytelling, music, computer science, etc...
- Thus games are designed *data-driven*
  - Game content is separated from game code
  - Examples:
    - Character data is kept in XML
    - Character behavior is specified through *scripting languages*
- Engine is reusable

Content Creation in Games



- Game software as artistic content
- Gameplay programmers versus software engineers versus designers

The Role Of Scripting Languages

Why scripting languages?

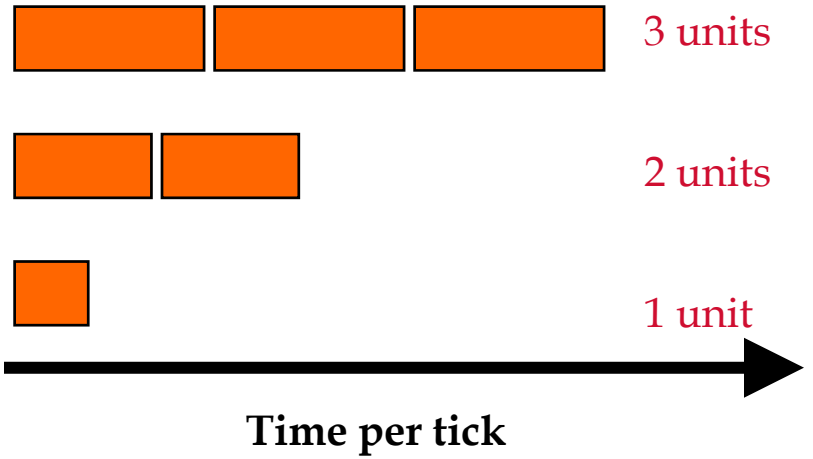
- Easy environment for gameplay programmers and designers
  - Sandbox for creation of "fun"
  - Make gameplay development fast and efficient
- User-created content (e.g., Second Life)
- Mods
  - Half-Life → Counter-Strike

Why Is Scaling NPCs Hard?

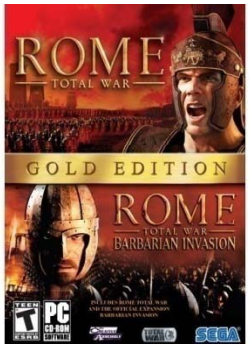
- Example: Morale
  - Battle between n knights and n skeletons
  - Assume knights that are afraid of skeletons
  - Morale inverse proportional to number of skeletons in view

Scaling NPCs (Contd.)

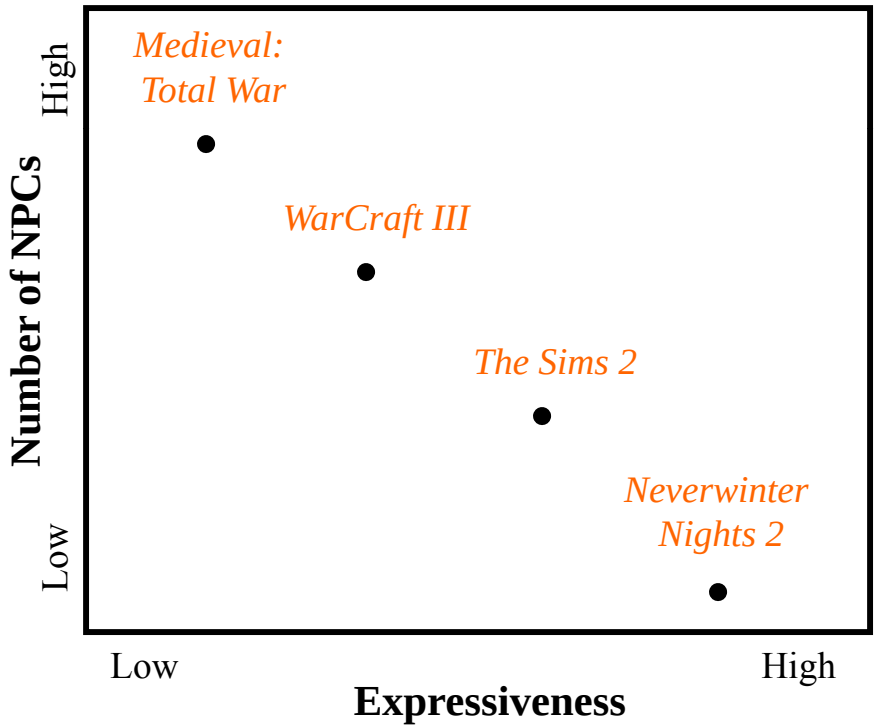
- Example: Morale
  - Battle between n knights and n skeletons
  - Assume knights that are afraid of skeletons
  - Morale inverse proportional to number of skeletons in view
- Each knight counts the number of skeletons in his view
  - O(n) per unit to count visible skeletons
  - O(n^2) to process all units
- Computation ≈ frame rate



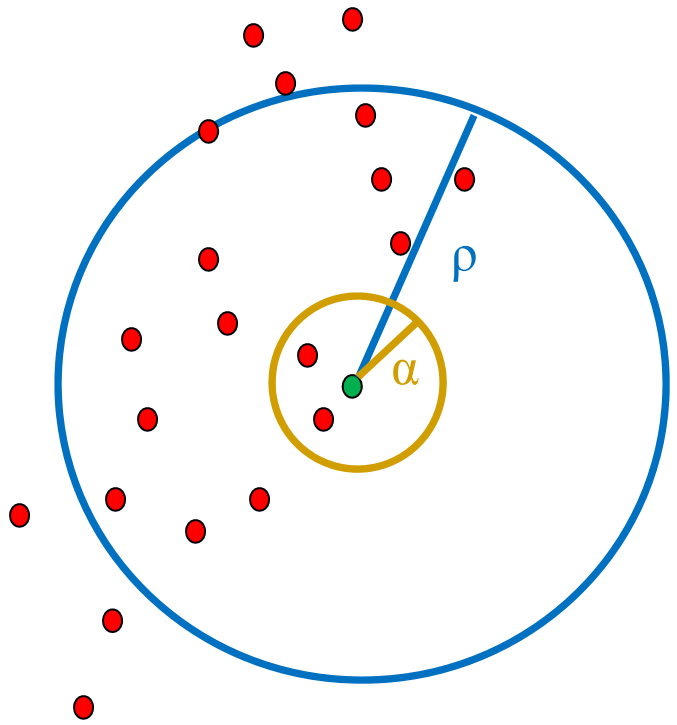
Expressiveness vs. Performance



- Expressiveness: The range of behavior that can be scripted (outside the engine).
- As the number of NPCs increases, expressiveness decreases.
  - *Neverwinter Nights 2*
    - Each NPC fully scriptable
  - *WarCraft III*
    - Script armies, not NPCs
    - Little NPC coordination
  - *Medieval: Total War*
    - No individual scripting at all



Scripting Simulations: Fish Schools



- Adapted from Couzin et al., Nature 2005
- Fish Behavior
  - Avoidance: if too close, repel other fish
  - Attraction: if within range, attract other fish

One Approach: Scripting in SGL

- High-level language
- Very similar to Java
- Programs specify behavior of individual simulated entities
- Syntax enforces the state-effect pattern
- Highly efficient execution through declarative processing

The State-Effect Pattern

- Objects in the game have attributes.
- The attributes are either states or effects.

States					Effects		
id	player	x	y	health	vx	vy	damage
1	1	12	342	100	0	0	0
2	1	43	12	100	0	0	0
3	2	123	90	95	0	0	0

Figure from Lineage II © NCSoft

The State-Effect Pattern

- State attributes are read only variables that are updated only at the end of a tick.
- Effect attributes are temporary variables that are used for intermediate computation during a tick.

States					Effects		
id	player	x	y	health	vx	vy	damage
1	1	12	342	100	0	0	0
2	1	43	12	100	0	0	0
3	2	123	90	95	0	0	0

Inside the Simulation Engine

A simulation tick has three phases:

States					Effects		
id	player	x	y	health	vx	vy	damage
1	1	12	342	100	0	0	0
2	1	43	12	100	0	0	0
3	2	123	90	95	0	0	0

Inside the Simulation Engine

A simulation tick has three phases:

- Query Phase
  - Read state variables
  - Compute values for effect variables. Multiple assignments to an effect are combined using a commutative and associative aggregation function.

States

id	player	x	y	health
1	1	12	342	100
2	1	43	12	100
3	2	123	90	95

Effects

vx	vy	damage
-10	3	
0	5	0
9	9	0

12
25
13

Cornell University

Inside the Simulation Engine

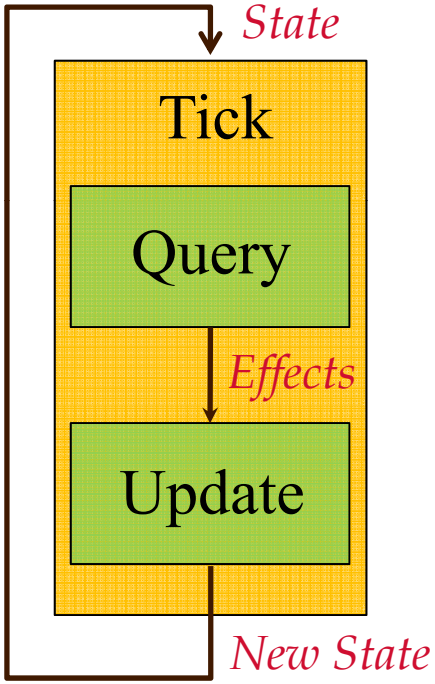
A simulation tick has three phases:

- Query Phase
  - Read state variables.
  - Compute values for effect variables. Multiple assignments to an effect are combined using a commutative and associative aggregation function.
- Update Phase
  - Compute new values for state variables from the effects and previous state variables.

States					Effects		
id	player	x	y	health	vx	vy	damage
1	1	2	345	50	0	0	0
2	1	43	17	100	0	0	0
3	2	132	99	95	0	0	0

Phases of a Tick

- Query
  - Reads State → Writes Effects
  - Each effect associated with associative-commutative combinator function
  - Effect writes order-independent
- Update
  - Reads Effects → Writes State
  - Each agent only reads its own state and effects
  - State writes order-independent



Another Example: Knights and Skeletons

```
class Skeleton {
   …
    public void run() {
      // Compute # of skeletons and center of crowd
      effect int c : sum, int sx : sum, int sy : sum;
      foreach (Skeleton f : Extent<Skeleton>)( {
        if (isEnemySkeleton(f) && dist(x,y,f.x,f.y < range) {
          c <- 1; sx <- f.x; sy <- f.y;
        }

      // If too many skeletons
      if (c > morale) {
         const int norm = (x-sx/c)*(x-sx/c)+ (y-sy/c)*(y-sy/c);
         // Run in opposite direction
         vx <- (x-sx/c)/norm; vy <- (y-sy/c)/norm;
      }
```

Cornell University

Another Example (Contd.)

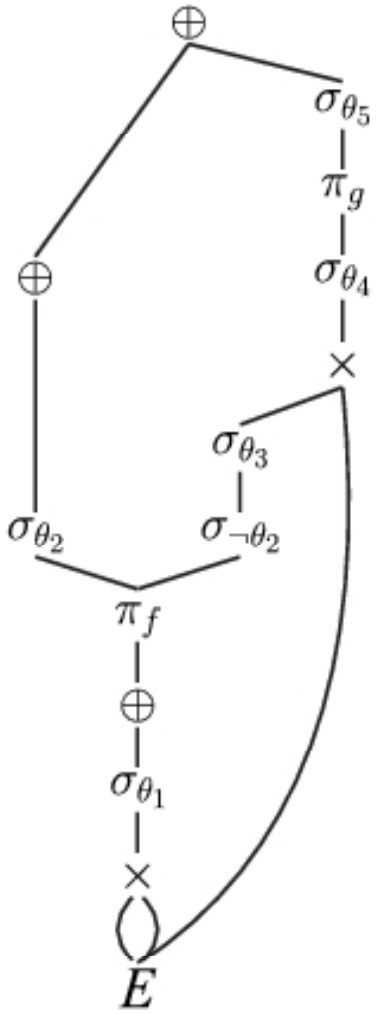
```
    if (c > morale) {
      . . .
    } else if (c > 0 && cooldown == 0) {
      // Find the nearest skeleton
     effect Skeleton target  : argmin (Skeleton s :
     dist(x,y,f.x,f.y,t));
     foreach  (Skeleton f: extent<Skeleton>) {
      if (isEnemySkeleton(f) { target <- r; }
     }
      // Attack it if found.
      if (target != null) { target.damage <- DMG_AMT; }
    }
  }// end void run()

  …

}// end class Skeleton
```

SGL Review

- SGL is an *imperative* language
- Users write programs for *individual* NPCs.
- Expressive power is limited so that SGL scripts can be compiled to Monad algebra
  - State-effect pattern
  - Restricted looping

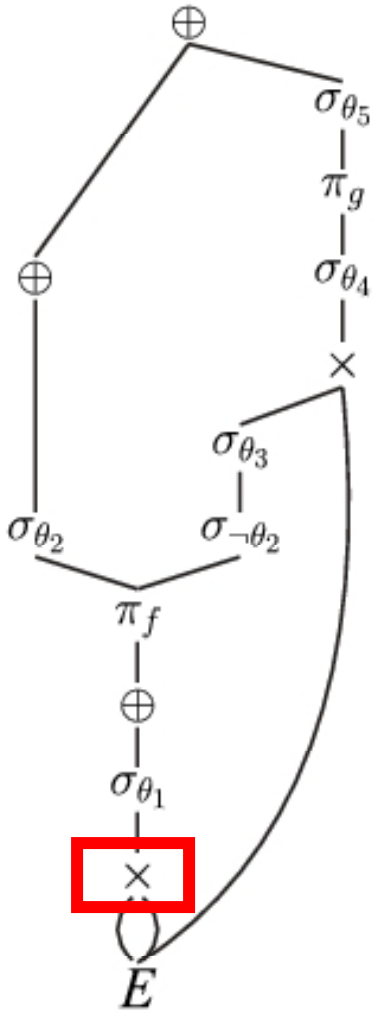
→ Declarative processing

Declarative Processing: An Example



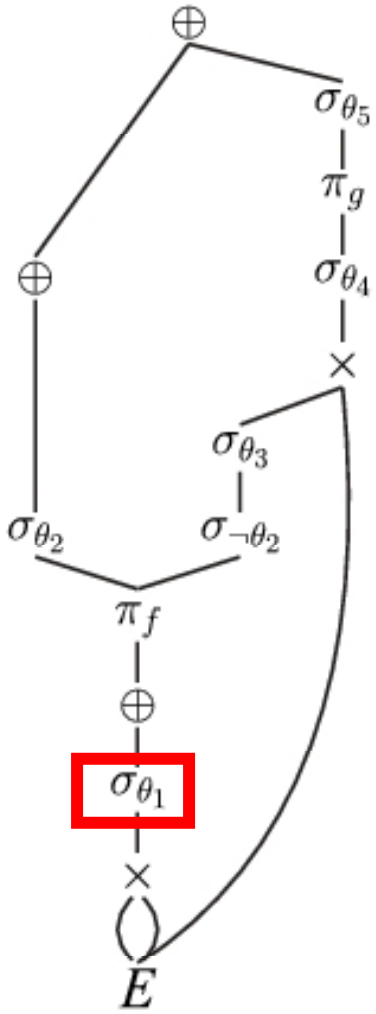

```
public void run() {
  // Compute # of skeletons and
  center of crowd
  effect int c : sum, int sx : sum,
  int sy : sum;
  foreach (Skeleton f :
  Extent<Skeleton>)( {
    if (isEnemySkeleton(f) &&
  dist(x,y,f.x,f.y < range) {
      c <- 1; sx <- f.x; sy <- f.y;
    }

  // If too many skeletons
  if (c > morale) {
     const int norm = (x-sx/c)*(x-
  sx/c)+ (y-sy/c)*(y-sy/c);
     // Run in opposite direction
     vx <- (x-sx/c)/norm; vy <- (y-
  sy/c)/norm;
}
```

Declarative Processing: An Example



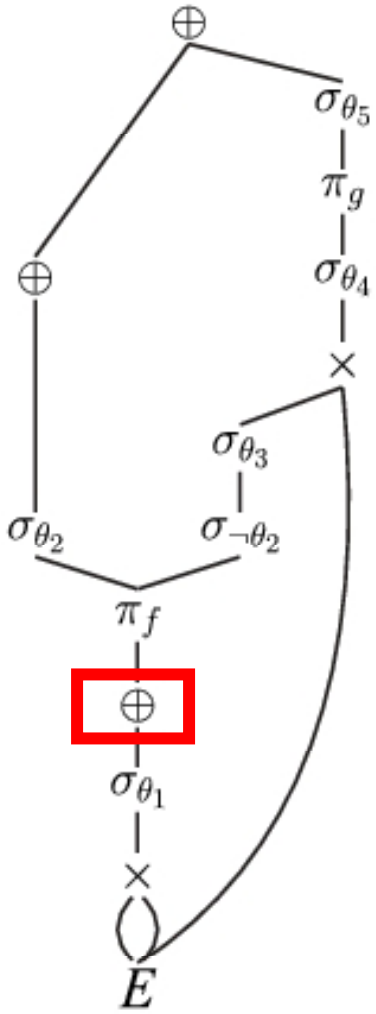

```
public void run() {
  // Compute # of skeletons and
  center of crowd
  effect int c : sum, int sx : sum,
  int sy : sum;
  foreach (Skeleton f :
  Extent<Skeleton>)( {
    if (isEnemySkeleton(f) &&
  dist(x,y,f.x,f.y < range) {
      c <- 1; sx <- f.x; sy <- f.y;
    }

  // If too many skeletons
  if (c > morale) {
    const int norm = (x-sx/c)*(x-
  sx/c)+ (y-sy/c)*(y-sy/c);
    // Run in opposite direction
    vx <- (x-sx/c)/norm; vy <- (y-
  sy/c)/norm;
}
```

Declarative Processing: An Example



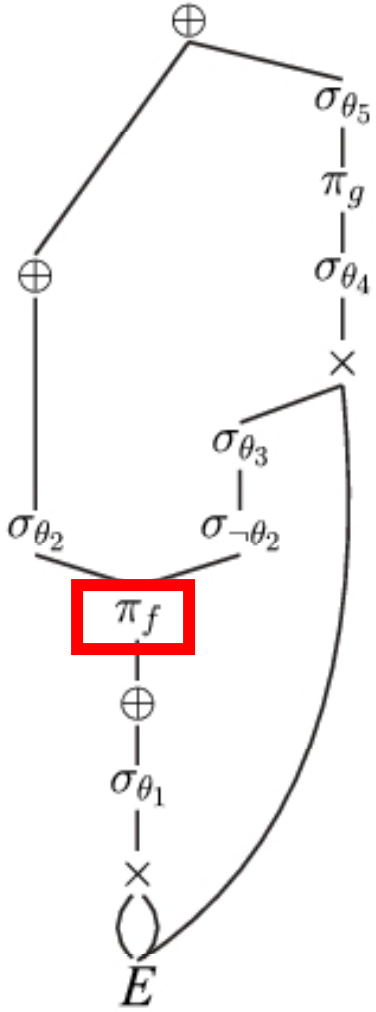

```
public void run() {
  // Compute # of skeletons and
  center of crowd
  effect int c : sum, int sx : sum,
  int sy : sum;
  foreach (Skeleton f :
  Extent<Skeleton>)( {
    if (isEnemySkeleton(f) &&
  dist(x,y,f.x,f.y < range) {
      c <- 1; sx <- f.x; sy <- f.y;
    }

  // If too many skeletons
  if (c > morale) {
    const int norm = (x-sx/c)*(x-
  sx/c)+ (y-sy/c)*(y-sy/c);
    // Run in opposite direction
    vx <- (x-sx/c)/norm; vy <- (y-
  sy/c)/norm;
}
```

Declarative Processing: An Example




```
public void run() {
  // Compute # of skeletons and
  center of crowd
  effect int c : sum, int sx : sum,
  int sy : sum;
  foreach (Skeleton f :
  Extent<Skeleton>)( {
    if (isEnemySkeleton(f) &&
  dist(x,y,f.x,f.y < range) {
      c <- 1; sx <- f.x; sy <- f.y;
    }

  // If too many skeletons
  if (c > morale) {
    const int norm = (x-sx/c)*(x-
  sx/c)+ (y-sy/c)*(y-sy/c);
    // Run in opposite direction
    vx <- (x-sx/c)/norm; vy <- (y-
  sy/c)/norm;
```

Declarative Processing: An Example




```
public void run() {
  // Compute # of skeletons and
  center of crowd
  effect int c : sum, int sx : sum,
  int sy : sum;
  foreach (Skeleton f :
  Extent<Skeleton>)( {
    if (isEnemySkeleton(f) &&
  dist(x,y,f.x,f.y < range) {
      c <- 1; sx <- f.x; sy <- f.y;
    }

  // If too many skeletons
  if (c > morale) {
     const int norm = (x-sx/c)*(x-
  sx/c)+ (y-sy/c)*(y-sy/c);
     // Run in opposite direction
     vx <- (x-sx/c)/norm; vy <- (y-
  sy/c)/norm;
  }
```

Code Optimizations

- Indexing
  - Construct an index in time $o(n^2)$
  - Lookup aggregate value through index nested-loops join
- Sweep-line algorithms
  - Compute all aggregate values in time $o(n^2)$ at the beginning of each tick

Simulations: Requirements

High-level programming

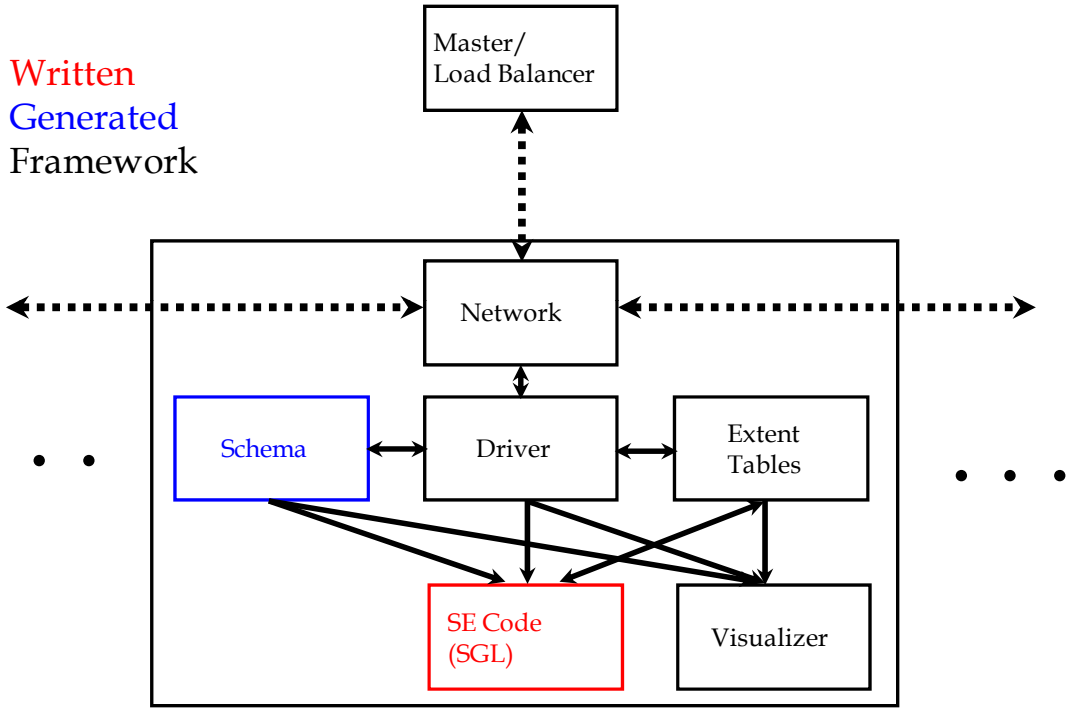
- SGL

Performance

- Compiles into C++, declarative processing

Scalability

- Automatic parallelization, completely transparent to the programmer



Cost-effectiveness

- Runs in the cloud
- Amazon EC2; get 1000 nodes for one hour for <$100

Automatic fault-tolerance

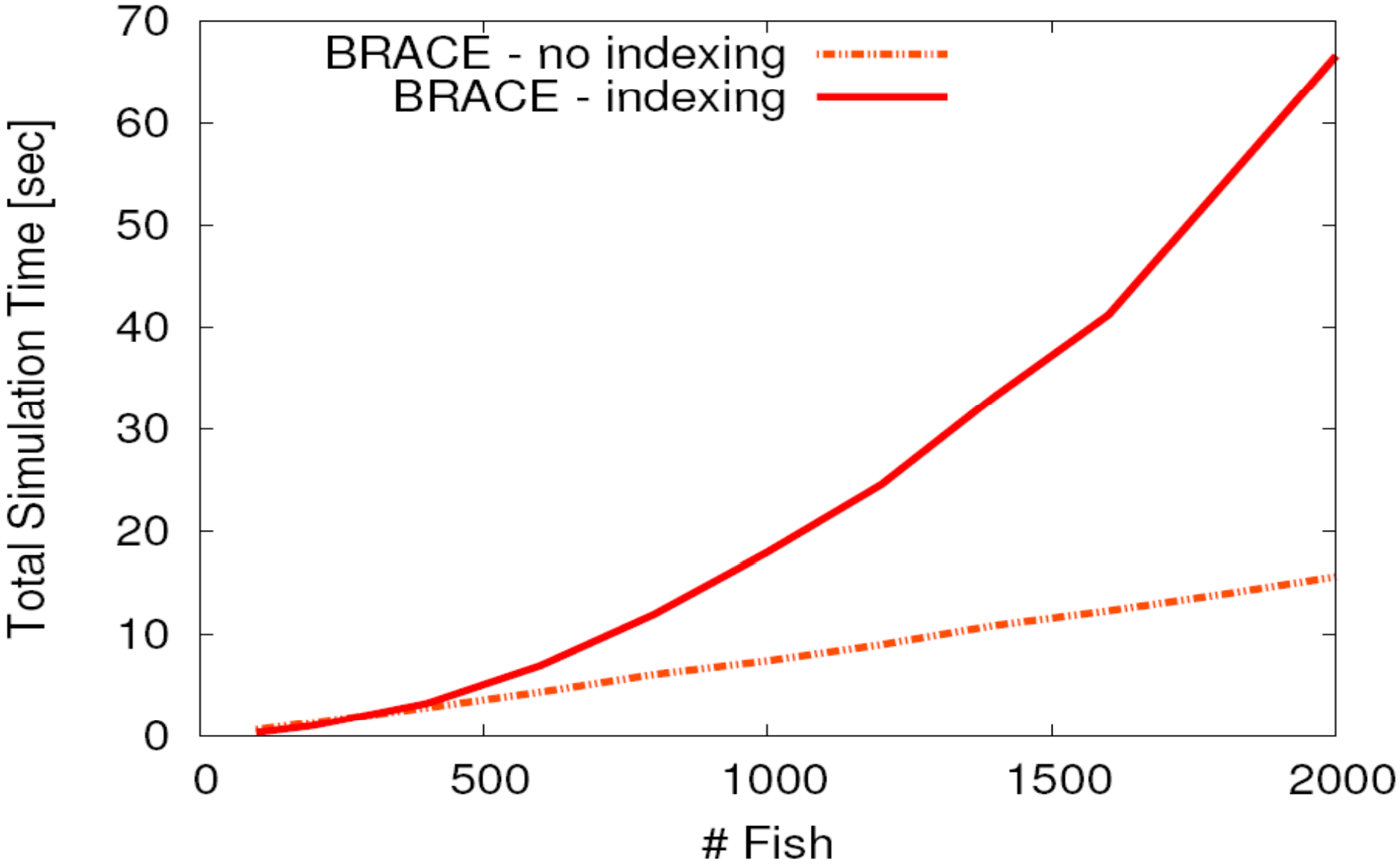
- Main-memory checkpointing techniques

Cornell University

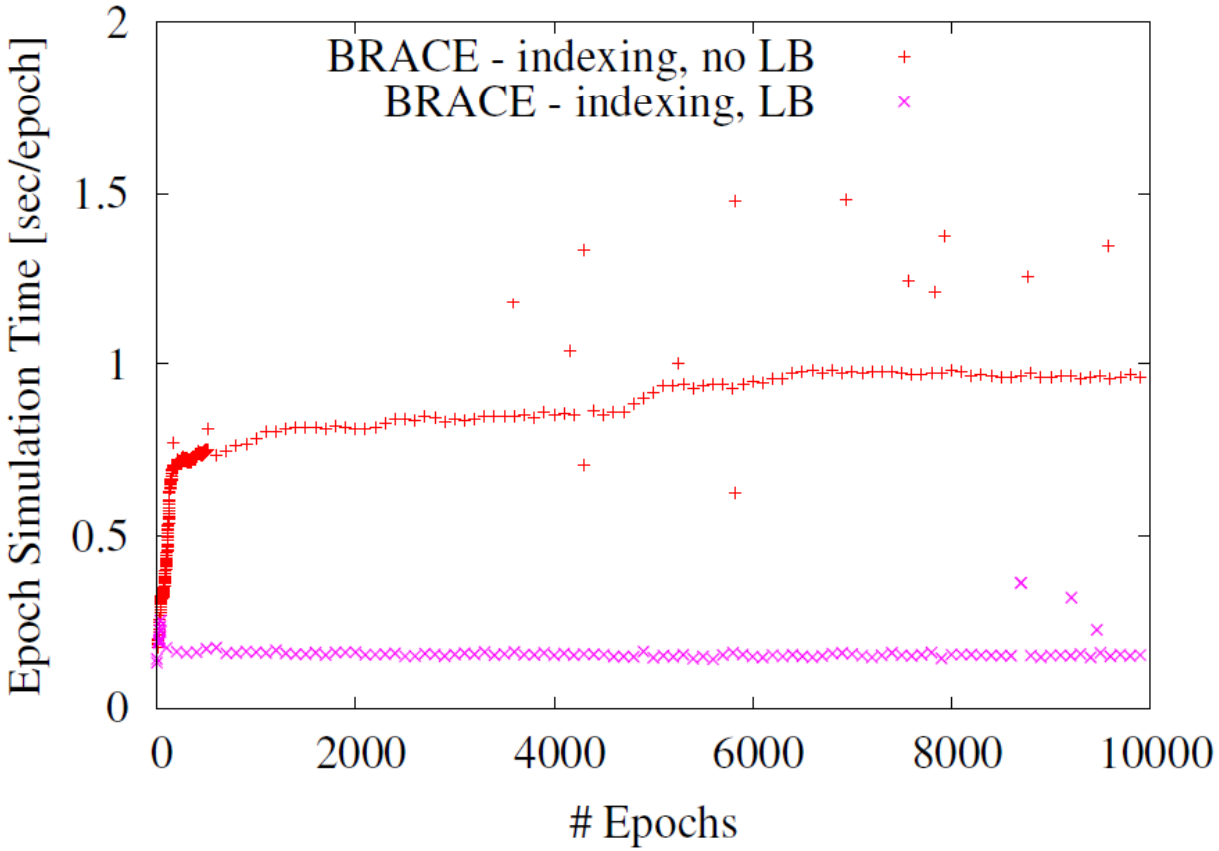
Example 1: Fish

[Couzin 2005]

Indexing of Fish Simulation



Load Balancing: Fish



- 16-node with load balancing turned on
- Fish simulation of two independent schools that swim in opposite directions

Example 2: Traffic

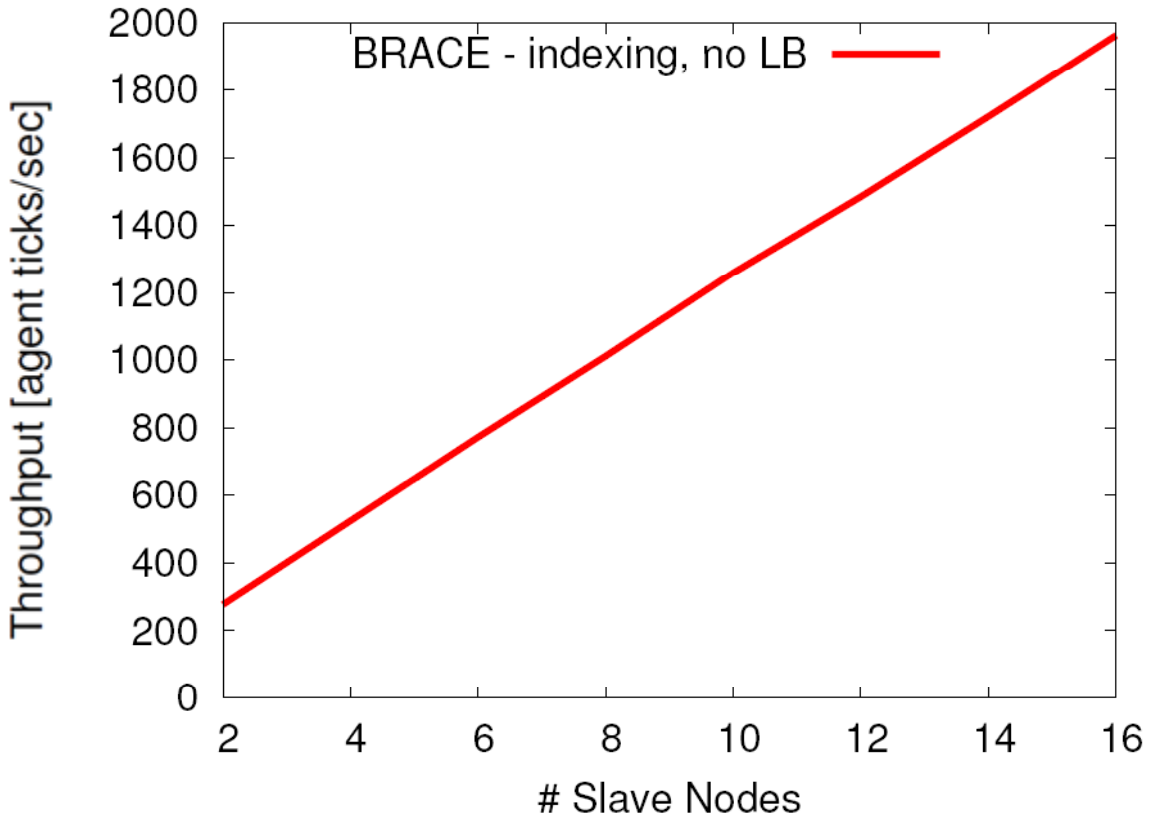
Source: EPA at http://www.epa.gov/

Example 2: Traffic (Contd.)

```
……
public void laneChange() {
    // get lead_gap
    effect number lead_gap : min;
    foreach Vehicle v in Extent<Vehicle> {
        if (v.link = this.link && v.lane = this.lane &&
            v.position > this.position)
                lead_gap <- v.position – this.position
                        – v.length/2 – this.length/2;
    }
     ……
    // get average_speed
    effect number num_cars : sum……
    effect number total_speed : sum……
    effect number average_speed = total_speed / num_cars;
    // compute utility
    effect number utility : priority
     ……
    if (this.lane – 1 > 0) // get left_lead, etc ……
    if (this.lane + 1 < getNumLane(this.link))
        // get right_lead, etc ……
        // set laneChange based on utility ……
}
```

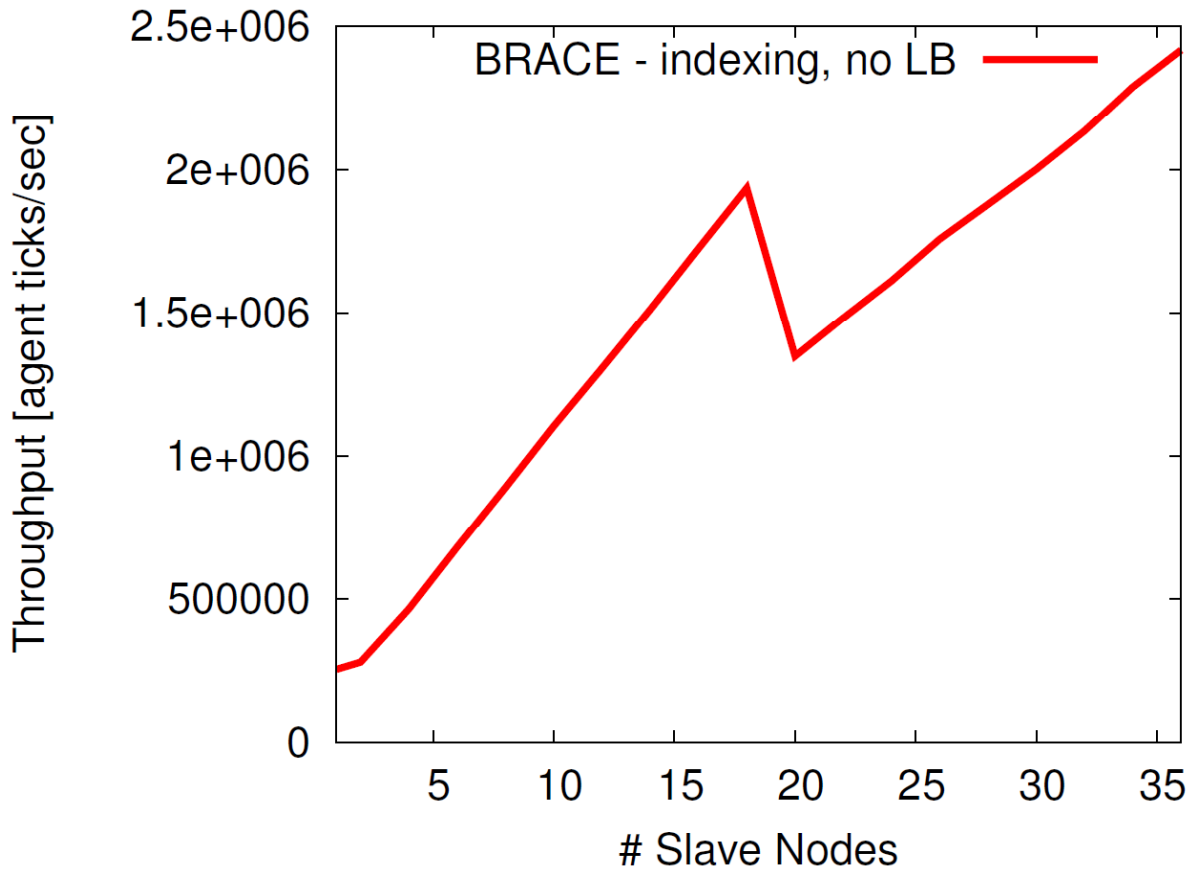
Cornell University

Traffic: Scalability



- Scale up of problem size with number of nodes
- Nearly linear scalability

Traffic: Scalability (Contd.)



- Scale up of problem size with number of nodes
- Nearly linear scalability

Summary: Declarative Processing

Novel way to program and process digital games and simulations

Main idea in this talk: Declarative processing

Much more: Automatic fault-tolerance, processing in the cloud, statistics collection, abstractions for building social games

One Remark: Creativity

- Engineers are creative!
- Games as an introduction to programming.
- Games as a way to express your creativity.

Meta Message

- Tools for data analysis are growing rapidly at a fast pace
- Ideas from database systems have wide applicability beyond managing data
- Capability of tools → level of creativity

Let's Play!

Questions?

johannes@cs.cornell.edu
http://www.cs.cornell.edu/johannes

Thank you: IARPA, National Science Foundation, Air Force Office of Scientific Research, New York City Metropolitan Transportation Council, Microsoft, Yahoo!, Intel, Google

Cornell University