

Resource Selection

Nut Taesombut
Guest Lecturer
Lecture #5

1

Resource Selection

- Last Time
 - » Resource Information
 - MDS: Grid Information Service
 - » Resource Description
 - Declarative Languages (RSL, ClassAds, Redline, SWORD)
 - Abstract, Application-Oriented Language (vgDL)
- Today
 - » Resource Selection
 - Matchmaking/Gang-matching
 - Heuristic Approaches with Domain Expert Knowledge
 - AI-Planning Approach
- Announcement
 - » Project Proposals are Due on Friday April 12, 2005

Today's Readings

- Raman et al, "Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching," HPDC-12, 2003.
- Huang et al, "Building Self-configuring Services Using Service-specific Knowledge," HPDC-13, 2004.
- Kichkaylo et al, "Optimal Resource-Aware Deployment Planning for Component-based Distributed Applications," HPDC-13, 2004.

=> Cover material from Thursday, April 7 lecture

Resource Selection

Resource Selection

- Given current resource state, find a set of resources that satisfy application needs
- Resource selection is a critical task
 - » Ensure applications achieve desired functionalities and quality of service
 - » Optimize application capabilities and performance
 - » Facilitate efficient utilization of system resources
 - Prevent excessive resource consumption
- Need an efficient resource selection scheme to maximize applications' and resource providers' benefits

Objectives

- Success and correctness
 - » Find a satisfying and realizable resource configuration
- Application capability optimization
 - » Based on application-specific criteria
 - E.g., maximizing link bandwidth for data-intensive applications
 - » Insert value-added components
- Global objective optimization
 - » Maximize system throughput
 - » Minimize resource consumption
 - » Achieve good load balancing

Why difficult ?

- Resource sharing across multiple administrative domains
 - » 10,000's or 100,000's of heterogeneous resources
 - » Dynamically changing resource behaviors
 - » Diverse resource usage policies
 - » Distributed/Incomplete view of resource information
- Complex application requirements
 - » Co-selection of multiple interdependent resources
 - » Wide range of constraints and preferences

Existing Resource Selection Schemes

- Attribute-based selection
 - » Condor's Matchmaking and Gangmatching
 - » Redline constraint solver
 - » SWORD
- Heuristic search with domain expert knowledge
 - » Recipe-based
- AI planning
 - » Sekitei

Condor Matchmaking

- Find a compatible match between a job classad and one resource classad
 - » Requirements <-> compatibility
 - » Rank <-> goodness
- Algorithm
 - » Iterate through the entire solution space
 - » Amongst all compatible classads, choose one with the highest "rank" value

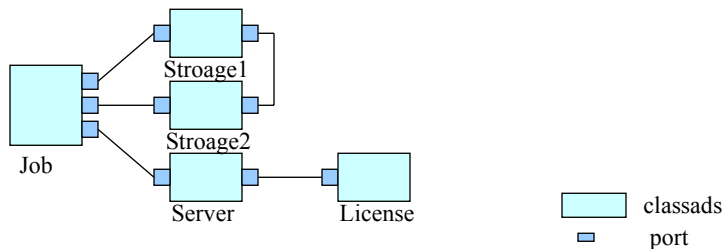
=> Guarantee optimality, but is limited to the selection of one resource

Example: A Compatible Match

```
[
  Type = "Job";
  Owner = "roy";
  Requirements = (other.OpSys == "Linux" && other.DiskSpaceMB >
  140);
  Rank = (other.DiskSpaceMB);
]
[
  Type = "Machine";
  OpSys = "Linux";
  DiskSpaceMB = 500;
  AllowedUsers = {"roy", "melski", "pfc"};
  Requirements = (IsMember(other.Owner, AllowedUsers);
```

Rank = 500

Gang-extended Matchmaker

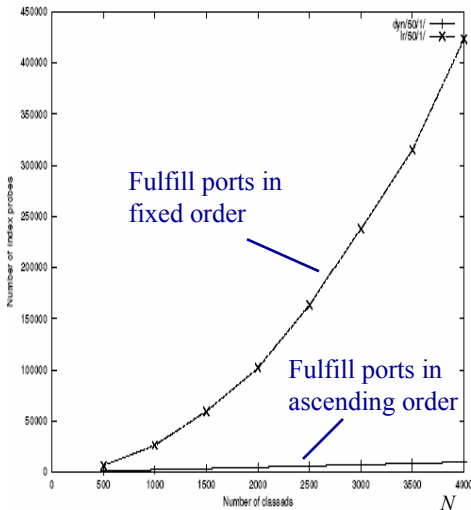


- Find a compatible match amongst a “gang” of classads
 - » Each includes a list of “ports” specifying requirements for matching classads

Gangmatching Algorithms

- Base Algorithm: Recursive backtracking search
 - » Start at a root classad (usually a job)
 - » Proceed sequentially to fulfill its ports by finding compatible classads
 - If a matching classad has unbound ports, designate it as a root and recursively fulfill its ports
 - If no match is found, backtrack to a previous port
 - » If all ports of the root are fulfilled, return with a valid solution
 - » If attempting to backtrack from a first port, return with no solution
- Optimization Techniques
 - » Use indexing scheme to exclude incompatible candidates
 - » Heuristic to fulfill ports in ascending order of candidate set size

Dynamic Vs. Fixed Order



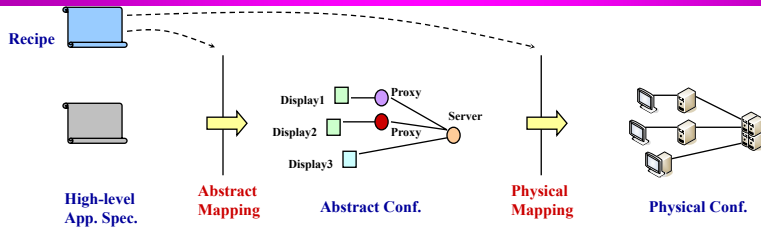
- Dynamic scheme
 - » Fulfill ports in ascending order
- Static scheme
 - » Fulfill ports in fixed order (left-to-right)
- A Job needs both a machine and a license
 - » N jobs, N machines and $N/2$ license
- Show # of indexes probed to satisfy $N/2$ jobs

13

Knowledge-based Resource Selection

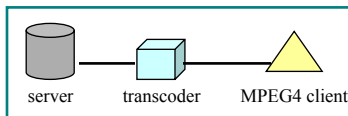
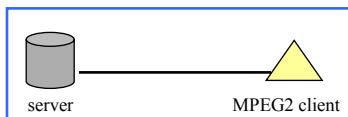
- Employ special knowledge to guide resource selection
 - » Heuristics/strategies for realizing good configurations
 - » Controllable optimization criteria
 - » Address complex/unusual constraints hard to declaratively express
 - Not simple attribute-based selection
- Two Schemes
 - » Domain Expert Knowledge
 - » General AI Planning Knowledge

Recipe-Based Application Composition (Cheng&Huang)



- Applications are dynamically built from distributed components
 - » Abstract mapping
 - Composition of application components into graph-like structure
 - Satisfying functional requirements
 - » Physical mapping
 - Mapping chosen components onto physical nodes
 - Satisfying resource requirements (e.g., bandwidth, memory, CPU speed)
- Use **application-specific knowledge (recipe)** to guide the mappings

Recipe



Abstract Mapping Knowledge

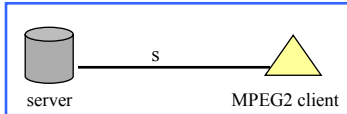
```

conf = new AbsConf()
server = conf.addComp("MPEG2Server")
if client.getProperty("VideoIn") == "MPEG2"
    conf.addConn(server, client)
else
    transcoder = conf.addComp("Transcoder")
    conf.addConn(server,transcoder)
    conf.addConn(transcoder,client)
    
```

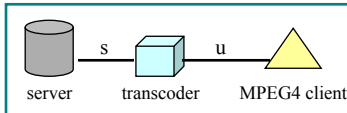
- Operational description of application-specific knowledge
 - » Abstract mapping knowledge
 - Determine which components and what connections are needed
 - » Physical mapping knowledge
 - Define objectives which drive heuristics (what're important!)

Recipe

Objective Function: $\min(s)$



Objective Function: $\min(s*M2 + u*M4)$



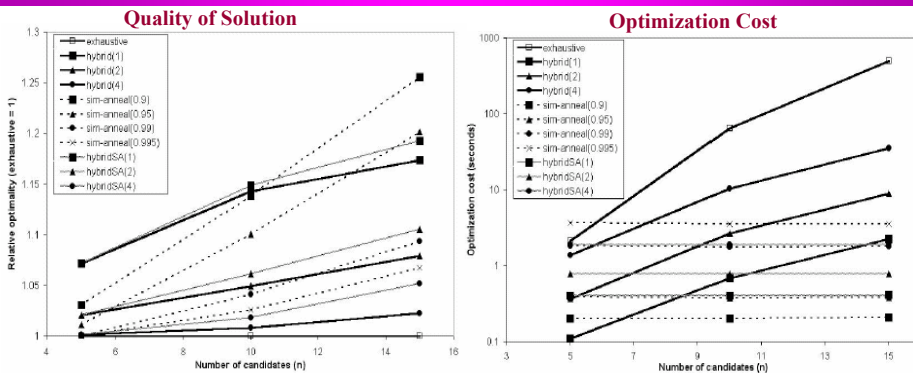
Physical Mapping Knowledge (in Blue)

```
conf = new AbsConf()
obj = new Term(0.0)
server = conf.addComp("MPEG2Server")
if client.getProperty("VideoIn") == "MPEG2"
  conf.addConn(server, client)
  obj.add(Latency(server, client))
else
  transcoder = conf.addComp("Transcoder")
  conf.addConn(server,transcoder)
  p_obj = new Term(Latency(server, transcoder))
  p_obj.multiplyBy(Double(MPEG2BitRate))
  obj.add(p_obj)
  conf.addConn(transcoder,client)
  p_obj = new Term(Latency(transcoder, client))
  p_obj.multiplyBy(Double(MPEG4BitRate))
  obj.add(p_obj)
objfunc = new Function(obj)
```

Realizing Physical Mapping

- Optimal Selection
 1. Exhaustive search
 - Iterate through the entire search space
 - » Guarantee optimality, but doesn't scale
- Heuristic Selection
 1. Simulated annealing
 - Randomized heuristic approach based on successive update steps
 2. Hybrid
 - Exhaustive search on candidate subsets (chosen based on local optimization)
 3. HybridSA
 - Simulated annealing on candidate subsets (chosen based on local optimization)
- » Control optimality/overhead tradeoff

Comparison of Different Heuristics

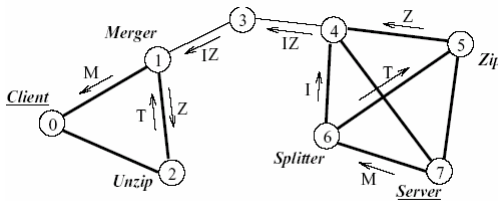


- Application composition for video conferencing
 - » 5 participants with n candidates
- Exhaustive and Hybrid offer good optimality, but their cost grows rapidly
- Simulated Annealing and HybridSA can achieve less optimality, but their cost remains relatively constant

Sekitei (Kichkaylo & Karamcheti)

- Applications are composed from modular components
 - » Assume components are reusable and dynamically deployable
- Sekitei is a generic planner
 - » Input: a description of environment, available components, and a desired goal
 - » Output: a deployment plan
 - » Combine the selection and placement of components
 - » Ability to express and manage a broad class of constraints
 - Qualitative: software, compatibility, authorization
 - Quantitative: CPU speed, link bandwidth

Application Deployment Problem



```

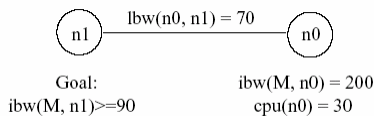
<component name=Merger>
<linkages>
  <requires>
    <interface name=T>
    <interface name=I>
  <implements>
    <interface name=M>
  <conditions>
    Node.cpu >= ( T.ibw+I.ibw )/5
    T.ibw*3 == I.ibw*7
  <effects>
    M.ibw := T.ibw + I.ibw
    Node.cpu -= ( T.ibw+I.ibw )/5
  
```

Goal: Deliver M stream from the server to the client

Components	Interface
Server	Media
Client	Text
Merger	Zipped Text
Splitter	Image
Zip	
Unzip	

- Fact of Life
- Nodes, links, and interfaces have real-valued properties
 - Function are non-reversible

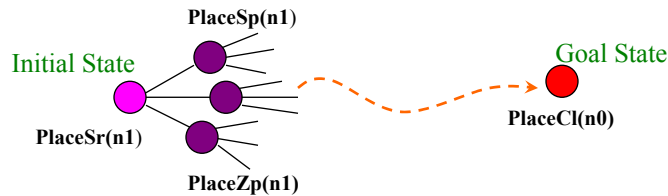
Sekitei's AI-planning Approach



- Deployment Plan**
- Place splitter on node n0
 - Place zip on node n0
 - Cross with Z stream from n0 to n1
 - Cross with I stream from n0 to n1
 - Place unzip on node n1
 - Place merger on node n1

- Problem domain is modeled as a set of world state and possible operators
 - » World State: current resource properties, and available interfaces on nodes
 - » Operators: deploying components, transfer an data stream
- Find a sequence of operators that produces a desired goal, starting from an initial state

Original Sekitei Algorithm



- Greedy Approach with Pruning Heuristic
 - » Consider only relevant operators for achieving the goal
 - » Assume functions are monotonic
 - » Propagate resource constraints from the initial state
 - Consider maximum utilization of resources
 - » Prefer a plan with minimum number of steps
- Shortcomings
 - » Inability to generate a valid plan in a resource-constrained environment
 - » Inability to specify the preferences on the generated plan

Sekitei Algorithm Extension

- Allow reasoning about reversibility through approximation
 - » Define discrete levels for resource property variables
 - » Mark these properties as being degradable, upgradeable, or neither
- Allow user-specified cost of an action
 - » Arbitrary function dependent of resource property values (what is important!)
 - » E.g., cost of “placing merger”: $1 + (I.ibw + T.ibw) / 10$
- Regression search guided by resource level/degradability information and estimated action costs
 - » Objective to minimize a sum of action costs (cost of a plan)

Evaluation of Sekitei Planner

- Key Questions
 - » Effect of resource-level on the quality of solution (cost and # of actions)
 - » Effect of resource-level on the planner performance (timing overhead and # of explored actions)
- Evaluation Configuration
 - » Application goal to deliver a media stream from the server to the client
 - » “Small” and “large” networks (6 and 93 nodes, respectively)
 - » Five resource-level scenarios

Scenario	Levels of bandwidth of M	Levels of link bandwidth
A	[0, ∞)	[0, ∞)
B	[0, 100), [100, ∞)	[0, ∞)
C	[0, 90), [90, 100), [100, ∞)	[0, ∞)
D	[0, 30), [30, 70), [70, 90), [90, 100), [100, ∞)	[0, ∞)
E	[0, 30), [30, 70), [70, 90), [90, 100), [100, ∞)	[0, 31), [31, 62), [62, ∞)

Result Part I: Quality of Solution

Scenario	Estimated Cost	# of Actions in Plan	Reserved BW
Small-A	No Solution	No Solution	No Solution
Small-B	10	10	100
Small-C	63	13	65
Small-D	63	13	65
Small-E	63	13	65
Large-A	No Solution	No Solution	No Solution
Large-B	11	11	100
Large-C	63	13	65
Large-D	63	13	65
Large-E	63	13	65

- The use of resource-level enables the planner to find a valid solution
- Finer-grained resource-level may help the finding of a better solution

Result Part II: Planner Performance

Scenario	# of Evaluated Actions	Search (msec)
Small-B	152	420
Small-C	222	291
Small-D	364	330
Small-E	1152	4366
Large-B	2528	3205
Large-C	3780	1041
Large-D	6198	671
Large-E	20386	25426

- Finer-grained resource-level can help detect unsatisfied constraints earlier
- However, it also increases # of actions to probe
- How should levels be specified to achieve an optimal solution?
 - » Optimality vs. Performance

Comparison

System Name	Input Specification	Assumption on Application	Selection Model	Special Knowledge	Optimization Goal
Gangmatching	Declarative Specification	Static Structure	Only Physical Mapping	None	Application Performance
Recipe-based	High-Level App. Goal	Dynamic Structure	Separate Abstract and Physical Mapping	Domain Expert Knowledge	Application Performance, Resource Utilization, or others
Sekitei	High-Level App. Goal	Dynamic Structure, and Small-Scale	Coordinated Abstract and Physical Mapping	Application-specific AI Planning Knowledge	Application Performance, Resource Utilization, or others

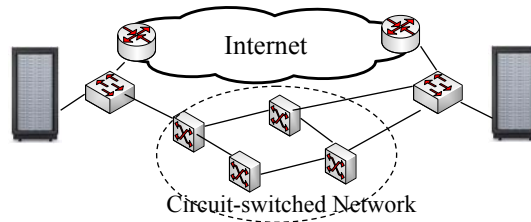
Other Challenges

29

Selection vs. Binding

- Race condition problem
 - » Resource sharing in wide-area distributed system
 - Applications compete to use resources
 - No centralized resource control/broker
 - » Chosen resources cannot be used as they are occupied by others
- Two models:
 - » Select and bind
 - Reserve or allocate resources during the selection
 - Prevent the race condition problem
 - Waste resources due to over-reservation
 - » Select, then bind
 - Provide more efficient resource utilization
 - May result in the race condition

Resource Selection in Grid with Configurable Networks



- Need select both end resources and network configurations
 - » Multi-path and qualitatively different networks
 - » Optimize applications for achieving a desired quality of connectivity (e.g., performance, reliability, etc.)
 - » Minimize the use of scarce network resources
 - Diverse network usage policies and information models
- => Significantly complicate the selection task

Different Selection Models

- Model I: Separate selection of end and network resources
 - » First, select a satisfying set of end resources
 - Ignore connectivity constraints at this step
 - » Then, choose applicable network configurations amongst them
- Model II: Coordinated selection
 - » Simultaneously select resources of both types
 - » Provide a greater degree of freedom in selection optimization

=> What benefits can we gain from coordinated selection?

Discussion

- Are these selection schemes practical in solving realistic problems? (scalability, supported applications, etc.)
- How would efficient representations of resource information and resource specification help?
- Can we have a single scheme that works well for all circumstances?
- If not, in which circumstances are these systems good for or likely to fail?
- How difficult to use these systems? What is the target level of users?

Summary

- Wide variety of resource selection schemes
- Gangmatching
 - » Generic attribute-based matching
 - » Indexing and search optimization techniques
- Recipe-based application composition
 - » Utilizing expert domain knowledge for component selection and placement (separate)
 - » Variety of search heuristics
- Sekitei
 - » AI-based planner for application composition
 - » Combine selection and placement of components