CS 516 Compilers and Programming Languages II

Redundancy-1
New Homework #3 deadline: Thursday, April 9.

Project #2 will be posted by this Friday/Saturday
Redundancy introduction

**redundant**: “Serving as a duplicate for preventing failure of an entire system (such as a spacecraft) upon failure of a single component.” (Merriam-Webster Dictionary, online edition)

Sample systems that use redundancy to improve system robustness

- **Check-sums** which may recover from single or multiple bit errors
- **Shannon’s theorem** (1948) describes the maximum possible efficiency of error-correcting methods under levels of noise and data corruption.
- **RAID** (Redundant Arrays of Inexpensive Disks - UC Berkeley)
- Five computers on the space shuttle (cloning with voting protocol)
- Multiple redundant software implementations with identical functionality
- Tandem computer company (now HP) that sells systems with identical components (cloning) and limited sharing
- Repeated execution: Check-pointing, error detection, and roll-back strategies, followed by re-execution

These approaches preserve the full system semantics, i.e., keep the system functionality the same in the presence of component failures.
Observation – Approximation can be considered a form of redundancy, allowing the notion of partial failure which produces outcomes of reduced quality

- Failure may be partial due to resource constraints.
- Partial system failure reduces the quality of the produced outcome, allowing for graceful quality degradation.
- Multiple system components can (partially) fail at the same time;
- A robust system may “force” some system component(s) to partially fail in order to allow other more crucial component(s) to remain functional, although at potentially reduced quality. This allows the overall system to “survive” at the highest efficiency/quality level given the available resource constraints.

Need for systems that can exploit traditional redundancy (cloning) and approximation by reconfiguring its components in response to resource limitations and/or component failures to achieve an overall optimal outcome.
Challenges:

• How to specify and build systems with redundancy and approximation?
• How to dynamically adapt the system to failure events and changing resource availabilities (runtime system)?
• How to define the notion of quality, which is an inherently subjective metric (part of system/application semantics)?
• How to quantitatively assess the notion of “robustness” for such systems, i.e., when is one system more robust than another?
• How to evaluate the benefit / success of such systems?
Basic assumptions and notions (ongoing research):

- An application / system consists of a set of interacting components, called services.
- Each service may have several approximation levels, with each level associated with a resource cost and quality.
- There are dependencies among services and their different approximation levels.
  - Redundancies may be expressed as alternative dependencies (OR dependencies).
  - Services may be cloned, together with their dependencies.
- A valid system state, called configuration, consists of the selection of a single approximation level for each required service.
- An optimal system configuration is a configuration that maximizes the overall system outcome quality while respecting a provided execution resource budget.
The RSDG represents single services (components) through layered nodes.

The single sample service \( S_1 \) has two approximation levels, where \( S_1^2 \leq S_1^1 \).

The highest level provides two implementations, A and B, that produce the same quality outcomes (functional redundancy).

Note: Cloning (identical copies) typically implies implementation redundancy, but implementation redundancy can also be achieved through providing the same functionality but using different resources.

Approximation and implementation redundancy
Redundant Services Dependence Graph (RSDG)

Edges in the RSDG represent dependencies among the different services and their approximation levels.

**AND dependencies** model a necessary selection of a “source” service and its service level when a “sink” service/level is selected.

**OR dependencies** allow alternative selections of services/levels. However, for each group of OR dependencies, exactly one “source” service has to be selected.
Services may be cloned by copying the service node and duplicating all dependencies, converting AND dependencies to OR dependencies from the set of clones.

$S_4a$ and $S_4b$ are identical clones of service $S_4$. Dependencies are duplicated.

Service cloning
The RSDG is a dually weighted, directed graph. The weights are cost ($C$ - resource consumption) and mission value ($Q$ - quality). For simplicity, we assume that all cost/quality models are linear.

Node and edge are associated with a cost $C$ (here energy). No label means 0 cost.

Node mission values and execution budgets are provided by the user. Mission values may only be given to user-critical services (nodes).

Basic structure of 0-1 problem:

$$\text{maximize } \sum_{i} \sum_{j} s_{ij} \cdot Q_{i}$$
$$\text{subject to } \sum_{j=1}^{n} s_{ij} \leq 1$$
$$\sum_{i} \sum_{j} C_{ij} \leq \text{Budget}$$
$$\text{foreach } \text{Edge}_{\text{AND}}(so \rightarrow si), si - so \leq 0$$
$$\text{foreach } \text{Edge}_{\text{OR}}(so_{1}, so_{2}, ..., so_{n} \rightarrow si), si - \sum_{i} so_{i} \leq 0$$
Redundant Services Dependence Graph (RSDG)

Example solutions for different provided resource budgets.

**Budget = 29**

Configuration = \{S^{1A}_1, S^{1}_3, S^{2}_2, S^{1}_5\}

Overall quality = 7
Overall energy cost = 29

**Budget = 31**

Configuration = \{S^{2}_1, S^{2}_3, S^{1}_2, S^{1}_4B\}

Overall quality = 9
Overall energy cost = 31

---

 cs516, spring 20 © U. Kremer  
 Lecture 19  
 11
Example solutions for different provided resource budgets.

What is the minimal cost configuration with the maximal mission value / quality?
Localization with five quality levels
- Two services with implementation redundancy
- One cloned service

Cost and quality weights are not shown implementation replication / cloning
Service failure inhibits services to be selected, and all other services that rely on the failed services.

Example: WiFi, Camera 2, and IMU 1 fail

⇒ limited configuration space

cost and quality weights are not shown
implementation replication / cloning
Possible metrics that describe the redundancy/robustness of a system

(1) Given a set of failed services FS (single and multiple failed services):
   • Reduction in configuration space size due to FS
   • Reduction in configuration selections of user critical services due to FS

(2) Sensitivity analysis to budget/resource reductions: Similar metrics can be used to assess the impact of budget reductions

(3) Impact of failures / limited budgets on quality of optimal configuration.
   • Enable “what if” experiments

⇒ Metrics can be used to assess and guide the design of robust/redundant systems