

---

# **DISTRIBUTED SYSTEMS: PARADIGMS AND MODELS**

**Academic Year 2010–2011**

---

**M. Danelutto**

**Teaching material – Laurea magistrale in Computer Science and Networking**



# CONTENTS

---

|  |           |
|--|-----------|
| Preface  | 1         |
| Acknowledgments  | 3         |
| <b>1 Parallel hardware<br/>(needs parallel software)</b> | <b>1</b>  |
| 1.1 Hardware for parallel/distributed computing          | 2         |
| 1.1.1 Evolution of CPUs                                  | 3         |
| 1.1.2 High performance computing architectures           | 6         |
| 1.1.3 Cloud  | 10        |
| 1.2 Parallel software urgencies                          | 11        |
| <b>2 Parallel programming: introducing the concept</b>   | <b>15</b> |
| 2.1 Concurrent activity graph                            | 18        |
| 2.2 Functional and non functional code                   | 19        |
| 2.3 Performance  | 19        |
| 2.4 Other non functional concerns                        | 21        |
| 2.4.1 Load balancing                                     | 21        |
| 2.4.2 Reliability  | 23        |
| 2.4.3 Security   | 24        |
| 2.4.4 Power management                                   | 25        |
| 2.5 “Classical” parallel programming models              | 25        |
| 2.5.1 POSIX/TCP  | 26        |
| 2.5.2 Command line interface                             | 28        |
| 2.6 Implementing parallel programs the “classical” way   | 29        |
| 2.6.1 The application                                    | 29        |
| 2.6.2 The concurrent activity graph                      | 29        |

|          |   |           |
|----------|---|-----------|
| 2.6.3    | Coordination  | 29        |
| 2.6.4    | Implementation  | 30        |
| 2.6.5    | Code relative to the implementation handling termination but not handling worker faults | 30        |
| 2.6.6    | Adding fault tolerance (partial)  | 37        |
| <b>3</b> | <b>Algorithmic skeletons</b>  | <b>45</b> |
| 3.1      | Algorithmic skeletons: definition(s)  | 46        |
| 3.1.1    | The skeleton advantage  | 48        |
| 3.1.2    | The skeleton weakness   | 50        |
| 3.2      | Skeletons as higher order functions with associated parallel semantics                  | 50        |
| 3.2.1    | Stream modelling  | 53        |
| 3.3      | A simple skeleton framework   | 54        |
| 3.3.1    | Stream parallel skeletons   | 54        |
| 3.3.2    | Data parallel skeletons   | 55        |
| 3.3.3    | Control parallel skeletons  | 59        |
| 3.3.4    | A skeleton framework  | 60        |
| 3.3.5    | Skeleton programs   | 61        |
| 3.4      | Algorithmic skeleton nesting  | 62        |
| 3.4.1    | Sequential code wrapping  | 63        |
| 3.4.2    | Full compositionality   | 65        |
| 3.4.3    | Two tier model  | 65        |
| 3.5      | Stateless vs. statefull skeletons   | 66        |
| 3.5.1    | Shared state: structuring accesses  | 67        |
| 3.5.2    | Shared state: parameter modeling  | 69        |
| <b>4</b> | <b>Implementation of algorithmic skeletons</b>  | <b>73</b> |
| 4.1      | Languages vs. libraries   | 73        |
| 4.1.1    | New language  | 73        |
| 4.1.2    | Library   | 74        |
| 4.2      | Template based vs. macro data flow implementation                                       | 76        |
| 4.2.1    | Template based implementations  | 76        |
| 4.2.2    | Macro Data Flow based implementation  | 82        |
| 4.2.3    | Templates vs. macro data flow   | 86        |
| 4.3      | Component based skeleton frameworks   | 86        |
|          | Problems  | 88        |
| <b>5</b> | <b>Performance models</b>   | <b>89</b> |
| 5.1      | Modeling performance  | 90        |
| 5.1.1    | “Semantics” associated with performance measures  | 91        |
| 5.2      | Different kind of models  | 92        |
| 5.3      | Alternative approaches  | 95        |
| 5.4      | Using performance models  | 96        |
| 5.4.1    | Compile time usage  | 97        |
| 5.4.2    | Run time usage  | 98        |

|          |  |            |
|----------|--|------------|
| 5.4.3    | Post-run time usage                              | 99         |
| 5.5      | Skeleton advantage                               | 100        |
| 5.6      | Monitoring application behaviour                 | 101        |
| 5.7      | Performance model design                         | 103        |
| 5.7.1    | Analytical performance models                    | 103        |
| 5.7.2    | Queue theory                                     | 104        |
| <b>6</b> | <b>Skeleton design</b>                           | <b>109</b> |
| 6.1      | Cole manifesto principles                        | 110        |
| 6.1.1    | Summarizing ...                                  | 112        |
| 6.2      | Looking for (new) skeletons                      | 112        |
| 6.2.1    | Analysis   | 112        |
| 6.2.2    | Synthesis  | 113        |
| 6.3      | Skeletons vs templates                           | 113        |
| <b>7</b> | <b>Template design</b>                           | <b>115</b> |
| 7.1      | Template building blocks                         | 117        |
| 7.1.1    | Client-server paradigm                           | 119        |
| 7.1.2    | Peer-to-peer resource discovery                  | 119        |
| 7.1.3    | Termination                                      | 120        |
| 7.2      | Cross-skeleton templates                         | 121        |
| 7.3      | Sample template mechanisms                       | 122        |
| 7.3.1    | Double/triple buffering                          | 122        |
| 7.3.2    | Time server                                      | 123        |
| 7.3.3    | Channel name server                              | 124        |
| 7.3.4    | Cache pre-fetching                               | 125        |
| 7.3.5    | Synchronization avoidance                        | 127        |
| 7.4      | Sample template design                           | 129        |
| 7.4.1    | Master worker template                           | 129        |
| 7.4.2    | Farm with feedback                               | 131        |
| <b>8</b> | <b>Portability</b>                               | <b>135</b> |
| 8.1      | Portability through re-compiling                 | 137        |
| 8.1.1    | Functional portability                           | 137        |
| 8.1.2    | Performance portability                          | 138        |
| 8.2      | Portability through virtual machines             | 139        |
| 8.3      | Heterogeneous architecture targeting             | 140        |
| 8.4      | Distributed vs multi-core architecture targeting | 142        |
| 8.4.1    | Template/MDF interpreter implementation          | 143        |
| 8.4.2    | Communication & synchronization                  | 144        |
| 8.4.3    | Exploiting locality                              | 144        |
| 8.4.4    | Optimizations                                    | 145        |
| <b>9</b> | <b>Advanced features</b>                         | <b>149</b> |
| 9.1      | Rewriting techniques                             | 149        |

|           |   |            |
|-----------|---|------------|
| 9.2       | Skeleton rewriting rules  | 151        |
| 9.3       | Skeleton normal form  | 152        |
|           | 9.3.1 Model driven rewriting  | 154        |
| 9.4       | Adaptivity  | 155        |
| 9.5       | Behavioural skeletons   | 160        |
|           | 9.5.1 Functional replication behavioural skeleton in GCM                      | 163        |
|           | 9.5.2 Hierarchical management   | 165        |
|           | 9.5.3 Multi concern management  | 170        |
|           | 9.5.4 Mutual agreement protocol   | 173        |
|           | 9.5.5 Alternative multi concern management                                    | 176        |
| 9.6       | Skeleton framework extendability  | 178        |
|           | 9.6.1 Skeleton set extension in template based frameworks                     | 179        |
|           | 9.6.2 Skeleton set extension through intermediate implementation layer access | 179        |
|           | 9.6.3 User-defined skeletons in <code>muskel</code>                           | 180        |
| <b>10</b> | <b>Skeleton semantics</b>   | <b>185</b> |
| 10.1      | Formal definition of the skeleton framework                                   | 185        |
| 10.2      | Operational semantics   | 186        |
| 10.3      | Parallelism and labels  | 190        |
| 10.4      | How to use the labeled transition system                                      | 191        |
| <b>11</b> | <b>Parallel design patterns</b>   | <b>193</b> |
| 11.1      | Skeletons   | 194        |
| 11.2      | Design patterns   | 194        |
| 11.3      | Skeletons vs parallel design patterns   | 195        |
| 11.4      | Skeletons $\cup$ parallel design patterns                                     | 197        |
| <b>12</b> | <b>Survey of existing skeleton frameworks</b>                                 | <b>199</b> |
| 12.1      | Classification  | 199        |
|           | 12.1.1 Abstract programming model features                                    | 199        |
|           | 12.1.2 Implementation related features  | 200        |
|           | 12.1.3 Architecture targeting features  | 201        |
| 12.2      | C/C++ based frameworks  | 201        |
|           | 12.2.1 P3L  | 201        |
|           | 12.2.2 Muesli   | 202        |
|           | 12.2.3 SkeTo  | 202        |
|           | 12.2.4 FastFlow   | 203        |
|           | 12.2.5 ASSIST   | 204        |
| 12.3      | Java based frameworks   | 204        |
|           | 12.3.1 Lithium/Muskel   | 204        |
|           | 12.3.2 Calcium  | 205        |
|           | 12.3.3 Skandium   | 205        |
| 12.4      | ML based frameworks   | 206        |
|           | 12.4.1 Skipper  | 206        |

|        |                                   |     |
|--------|-----------------------------------|-----|
| 12.4.2 | OcamlP3L                          | 206 |
| 12.5   | Component based frameworks        | 207 |
| 12.5.1 | GCM Behavioural skeletons         | 207 |
| 12.5.2 | LIBERO                            | 208 |
| 12.6   | <i>Quasi</i> -skeleton frameworks | 208 |
| 12.6.1 | TBB                               | 208 |
| 12.6.2 | TPL                               | 209 |
| 12.6.3 | OpenMP                            | 209 |
|        | List of Figures                   | 211 |
|        | List of Tables                    | 215 |
|        | Acronyms                          | 217 |
|        | Index                             | 219 |
|        | References                        | 223 |