

Table of Contents

Preface	VII
1. Introduction	1
1.1 I/O Requirements of Parallel Scientific Applications	2
1.1.1 Earthquake Ground Motion Modeling	5
1.1.2 Image Analysis of Planetary Data	5
1.1.3 Climate Prediction	6
1.1.4 Summary	6
1.2 State-of-the-Art and Statement of the Problems	6
1.3 The Central Problem Areas Addressed in this Book	11
2. Parallel Programming Models	13
2.1 The Parallel Machine Model	13
2.1.1 IMPA Model	14
2.1.2 Matching the IMPA to Real Architectures	16
2.2 Programming Models for Distributed-Memory Systems	20
2.2.1 Message-Passing Programming Model	21
2.2.2 Data Parallel Programming Model	22
2.2.3 Models Based on Coordination Mechanisms	24
Bibliographical Notes	24
3. Vienna Fortran 90 and Its Extensions for Parallel I/O	25
3.1 Language Model	25
3.1.1 Index Domain and Index Mapping	26
3.1.2 Index Domain of an Array Section	27
3.1.3 Data Distribution Model for Internal Memory	27
3.2 Language Constructs	29
3.2.1 Distribution Annotations	29
3.2.2 Alignment Annotations	32
3.2.3 Dynamic Distributions and the DISTRIBUTE State- ment	32
3.2.4 Indirect Distribution	33
3.2.5 Procedures	35
3.2.6 FORALL Loop	36

3.2.7	Reduction Operators	37
3.2.8	User-Defined Distribution Functions	38
3.2.9	Support for Sparse Matrix Computation	38
3.3	Controlling Parallel I/O Operations	42
3.3.1	The File Processing Model	43
3.3.2	User's Perspective	45
3.3.3	Data Distribution Model for External Memory	46
3.3.4	Opening a Parallel File	49
3.3.5	Write and Read Operations on Parallel Files	58
3.3.6	I/O Alignment	60
3.3.7	Other I/O Statements	60
3.3.8	Intrinsic Procedures	60
3.3.9	Experiments	61
3.4	Out-of-Core Annotation	68
3.4.1	User Controlled Mode	69
3.4.2	Automatic Mode	72
	Bibliographical Notes	73
4.	Compiling In-Core Programs	75
4.1	Parallelizing Compilation Systems	75
4.2	Parallelization Strategies for Regular Codes	79
4.2.1	Parallelization of Loops	80
4.2.2	Adaptation of Array Statements	91
	Bibliographical Notes	98
4.3	Parallelization of Irregular Codes	100
4.3.1	Irregular Code Example: Computation on an Unstructured Mesh	101
4.3.2	Programming Environment for Processing Irregular Codes	103
4.3.3	Working Example	106
4.3.4	Distribution Descriptors	107
4.3.5	Physical Data Redistribution	110
4.3.6	Work Distributor, Inspector and Executor	110
4.3.7	Handling Arrays with Multi-Dimensional Distributions and General Accesses	116
4.3.8	Runtime Support	120
4.3.9	Optimizations	120
4.3.10	Performance Results	123
	Bibliographical Notes	124
4.4	Coupling Parallel Data and Work Partitioners to the Compiler	125
4.4.1	Towards Parallel Data and Work Partitioning	128
4.4.2	Partitioners of the CHAOS Runtime Library	130
4.4.3	High Level Language Interface	131
4.4.4	Interactive Specification of the Partitioning Strategy ..	133
4.4.5	Implementation Issues	133

4.4.6	Illustration of the Approach	135
4.4.7	Performance Results	140
	Bibliographical Notes	142
4.5	Compile Time Optimizations for Irregular Codes	142
4.5.1	Introduction	142
4.5.2	Partial Redundancy Elimination	144
4.5.3	Specifiers of the Candidates for Placement	147
4.5.4	Influencers of the Candidate for Placement	149
4.5.5	The Form of Specifiers	150
4.5.6	Intraprocedural Optimizations	153
4.5.7	Interprocedural Optimizations	163
	Bibliographical Notes	167
5.	Compiling Parallel I/O Operations	169
5.1	Direct I/O Compilation	170
5.1.1	OPEN Statement	170
5.1.2	WRITE Statement	172
5.1.3	READ Statement	174
5.2	Optimizing I/O Compilation	174
5.2.1	I/O Communication Descriptors	176
5.2.2	Automatic Determination of I/O Distribution Hints ..	177
5.2.3	Overlapping I/O with Computation	178
5.2.4	Running I/O Concurrently with Computation and Other I/O	178
	Bibliographical Notes	181
6.	Compiling Out-of-Core Programs	183
6.1	Models for Out-of-Core Computations	184
6.1.1	Local Placement Model	186
6.1.2	Global Placement Model	187
6.1.3	Partitioned In-core Model	189
6.1.4	Model Based on a Parallel Array Database	190
6.2	Compilation Strategy	191
6.3	Specification of the Application Area	192
6.4	Compiling Out-of-Core Regular Codes	193
6.4.1	Out-of-Core Restructuring Techniques Applied to Loosely Synchronous Computations	193
6.4.2	Out-of-Core Restructuring Techniques Applied to Pipelined Computations	197
6.4.3	Optimizations for Out-of-Core Regular Codes	197
6.5	Parallelization Methods for Out-of-Core Irregular Problems ..	202
6.5.1	Problem Description and Assumptions	203
6.5.2	Data Arrays are In-Core and Indirection/Interaction Arrays Out-of-Core	205
6.5.3	Experiments	214

6.5.4	Generalization of Irregular Out-of-Core Problems	217
6.6	Support for I/O Oriented Software-Controlled Prefetching . . .	226
6.6.1	Compiler-Directed Data Prefetching	227
6.6.2	Data Prefetching Based on Prefetch Adaptivity	229
	Bibliographical Notes	233
7.	Runtime System for Parallel Input-Output	235
7.1	Coupling the Runtime System to the VFCS	235
7.2	Data Mapping Model	237
7.2.1	Mapping Data Arrays to Computing Processors	238
7.2.2	Logical Storage Model	238
7.2.3	Physical Storage Model	239
7.2.4	Example of the Model Interpretation	240
7.3	Design of the VIPIOS	241
7.3.1	Design Objectives	241
7.3.2	Process Model	242
7.4	Data Locality	245
7.4.1	Logical Data Locality	245
7.4.2	Physical Data Locality	245
7.5	Two-Phase Data Administration Process	245
7.5.1	Preparation Phase	246
7.5.2	Administration Phase	246
7.6	Basic Execution Profile	246
7.7	VIPIOS Servers - Structure and Communication Scheme . . .	250
7.7.1	Handling I/O Requests - Control Flow and Data Flow in the VIPIOS	252
7.7.2	Implementation Notes	254
	Bibliographical Notes	254
8.	A New Generation Programming Environment for Parallel Architectures	257
8.1	Overview	257
8.2	Compilation and Runtime Technology	258
8.3	Debugging System	260
8.3.1	Motivation	260
8.3.2	Design of a High-Level Symbolic Debugger	261
	Bibliographical Notes	266
	References	267
	Subject Index	283