

DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language

Microsoft Research Silicon Valley ¹joint affiliation, Reykjavik University, Iceland

Abstract

DryadLINQ is a system for distributed data-parallel computing using a high-level language. It is designed to be easy to use and to support a wide range of applications. DryadLINQ is based on the Dryad system, which is a distributed data-parallel computing system. DryadLINQ extends Dryad with a high-level language, LINQ, which allows users to write programs in a high-level language that is easy to use and to understand. DryadLINQ is designed to be easy to use and to support a wide range of applications. DryadLINQ is based on the Dryad system, which is a distributed data-parallel computing system. DryadLINQ extends Dryad with a high-level language, LINQ, which allows users to write programs in a high-level language that is easy to use and to understand.

1 Introduction

DryadLINQ is a system for distributed data-parallel computing using a high-level language. It is designed to be easy to use and to support a wide range of applications. DryadLINQ is based on the Dryad system, which is a distributed data-parallel computing system. DryadLINQ extends Dryad with a high-level language, LINQ, which allows users to write programs in a high-level language that is easy to use and to understand.

DryadLINQ is a system for distributed data-parallel computing using a high-level language. It is designed to be easy to use and to support a wide range of applications. DryadLINQ is based on the Dryad system, which is a distributed data-parallel computing system. DryadLINQ extends Dryad with a high-level language, LINQ, which allows users to write programs in a high-level language that is easy to use and to understand. DryadLINQ is designed to be easy to use and to support a wide range of applications. DryadLINQ is based on the Dryad system, which is a distributed data-parallel computing system. DryadLINQ extends Dryad with a high-level language, LINQ, which allows users to write programs in a high-level language that is easy to use and to understand.

15, 26, 31, 34, 35

14

15

E

32

31

F H E

H

D

D

virtualized

25

21, 24, 27, 28, 29

A

D

D

2

D

D

4

D

5

6

7

2 System Architecture

D

26

A D

A

D

E

(1)

(2)

(3)

(4)

(5)

A

D

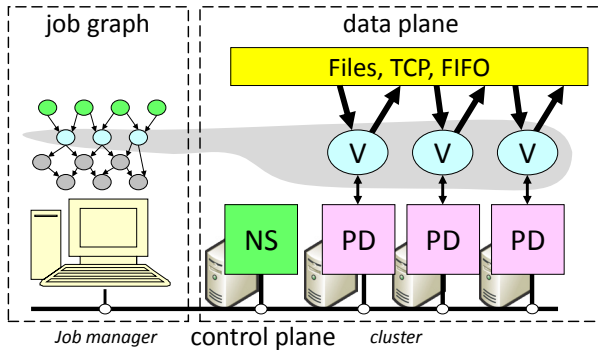


Figure 1: Dryad system architecture. NS is the name server which maintains the cluster membership. The job manager is responsible for spawning vertices (V) on available computers with the help of a remote-execution and monitoring daemon (PD). Vertices exchange data through files, TCP pipes, or shared-memory channels. The grey shape indicates the vertices in the job that are currently running and the correspondence with the job execution graph.

2.1 DryadLINQ Execution Overview

Figure 2: LINQ-expression execution in DryadLINQ.

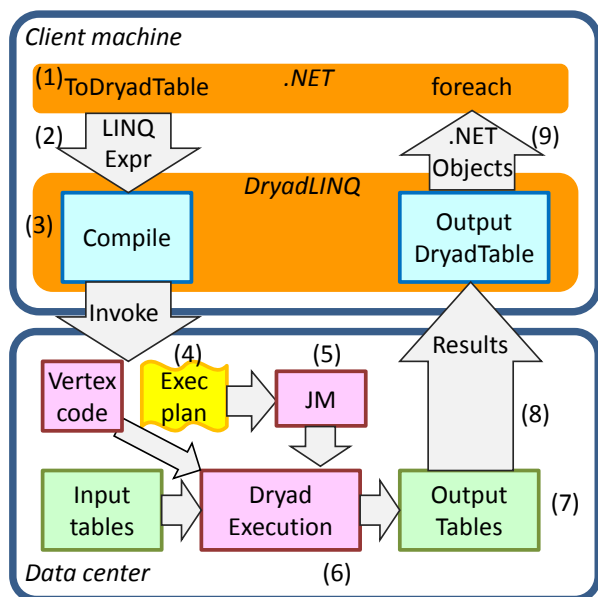


Figure 2: LINQ-expression execution in DryadLINQ.

Step 1. A .NET Expression E is passed to ToDryadTable (Figure 3).

Step 2. ToDryadTable calls Compile on DryadLINQ.

Step 3. DryadLINQ calls Compile on DryadExecution (Figure 4).

DryadExecution (Figure 5).
Step 4. DryadExecution calls DryadExecution (Figure 6).

Step 5. DryadExecution calls DryadExecution (Figure 7).

Step 6. DryadExecution calls DryadExecution (Figure 8).

Step 7. DryadExecution calls DryadExecution (Figure 9).

Step 8. DryadExecution calls DryadExecution (Figure 10).

Step 9. DryadExecution calls DryadExecution (Figure 11).

Step 10. DryadExecution calls DryadExecution (Figure 12).

Step 11. DryadExecution calls DryadExecution (Figure 13).

Step 12. DryadExecution calls DryadExecution (Figure 14).

3 Programming with DryadLINQ

DryadLINQ is a .NET library that allows you to write LINQ expressions that are executed on a Dryad cluster. The library is designed to be easy to use and to integrate with the .NET ecosystem. It provides a set of classes and methods that allow you to write LINQ expressions that are executed on a Dryad cluster. The library is designed to be easy to use and to integrate with the .NET ecosystem.

3.1 LINQ

2. The DryadLINQ library provides a set of classes and methods that allow you to write LINQ expressions that are executed on a Dryad cluster. The library is designed to be easy to use and to integrate with the .NET ecosystem. It provides a set of classes and methods that allow you to write LINQ expressions that are executed on a Dryad cluster. The library is designed to be easy to use and to integrate with the .NET ecosystem.

```

public interface IQueryable<T>
    IEnumerable<T>
    {
        // ...
    }

public interface IQueryable
    IQueryable<T>
    {
        // ...
    }

public class QueryScoreDocIDTriple
    {
        // ...
    }

var adjustedScoreTriples =
    from d in scoreTriples
    join r in staticRank on d.docID equals r.key
    select new QueryScoreDocIDTriple(d, r);

var rankedQueries =
    from s in adjustedScoreTriples
    group s by s.query into g
    select TakeTopQueryResults(g);

// Object-oriented syntax for the above join
var adjustedScoreTriples =
    scoreTriples.Join(staticRank,
        d => d.docID, r => r.key,
        (d, r) => new QueryScoreDocIDTriple(d, r));

var groupedQueries =
    adjustedScoreTriples.GroupBy(s => s.query);

var rankedQueries =
    groupedQueries.Select(
        g => TakeTopQueryResults(g));

```

```

// SQL-style syntax to join two input sets:
// scoreTriples and staticRank
var adjustedScoreTriples =
    from d in scoreTriples
    join r in staticRank on d.docID equals r.key
    select new QueryScoreDocIDTriple(d, r);
var rankedQueries =
    from s in adjustedScoreTriples
    group s by s.query into g
    select TakeTopQueryResults(g);

// Object-oriented syntax for the above join
var adjustedScoreTriples =
    scoreTriples.Join(staticRank,
        d => d.docID, r => r.key,
        (d, r) => new QueryScoreDocIDTriple(d, r));
var groupedQueries =
    adjustedScoreTriples.GroupBy(s => s.query);
var rankedQueries =
    groupedQueries.Select(
        g => TakeTopQueryResults(g));

```

Figure 3: A program fragment illustrating two ways of expressing the same operation. The first uses LINQ's declarative syntax, and the second uses object-oriented interfaces. Statements such as `r => r.key` use C#'s syntax for anonymous lambda expressions.

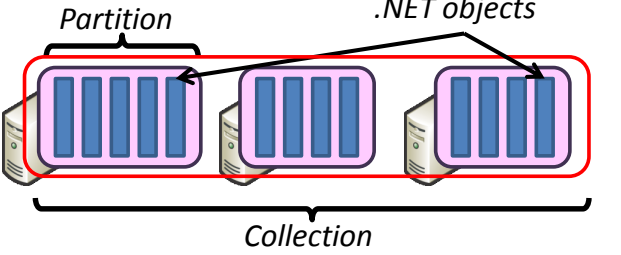


Figure 4: The DryadLINQ data model: strongly-typed collections of .NET objects partitioned on a set of computers.

3.2 DryadLINQ Constructs

```

public interface DryadTable<T>
    IQueryable<T>
    {
        // ...
    }

public class DryadTable<T>
    DryadTable<T>
    {
        // ...
    }

```

```

public interface DryadTable<T>
    IQueryable<T>
    {
        // ...
    }

public class DryadTable<T>
    DryadTable<T>
    {
        // ...
    }

```

```

4.
    Df
    all the func-
    tions called in DryadLINQ expressions must be side-
    effect free.
    H
    Df
    Df
    GetTable<T>
    ToDryadTable<T>

var input = GetTable<LineRecord>("file://in.tbl");
var result = MainProgram(input, ...);
var output = ToDryadTable(result, "file://out.tbl");

```

```

    Df
    ToDryadTable
    DryadTable
    Df
    (
    )
    Df
    HashPartition<T,K>
    RangePartition<T,K>
    Ff
    Apply
    Fork
    Apply
    f
    A
    Apply
    [i, i + d]
    Apply
    7.
    Fork
    Apply

```

```

    Df
    f, Apply ← Fork /
    f
    Apply
    Apply
    Df
    Df
    A
    Apply
    E
    Resource
    Df
    Apply

```

3.3 Building on DryadLINQ

```

    Df
    C#
    15
    (
    ):
    public static MapReduce( // returns set of Rs
        source, // set of Ts
        mapper, // function from T → Ms
        keySelector, // function from M → K
        reducer // function from (K,Ms) → Rs
    ) {
        var mapped = source.SelectMany(mapper);
        var groups = mapped.GroupBy(keySelector);
        return groups.SelectMany(reducer);
    }

```

```

4
D#
5
14;
D# C#
38

```

4 System Implementation

```

D#
3
2
D#
25
D#

```

4.1 Execution Plan Graph

```

D#
(E G)
E G
E G
Fork
E G
D#
E G
E G
D#
E G
(
E G
E
F#

```

```

OrderBy
4.2.3.
E G
D#
G
input.Select(x => f(x)). f
x.name,
D#
H
f
F#
D#

```

4.2 DryadLINQ Optimizations

```

D#
D#

```

4.2.1 Static Optimizations

```

D#
E G

```

Pipelining:

```


```

Removing redundancy:

```

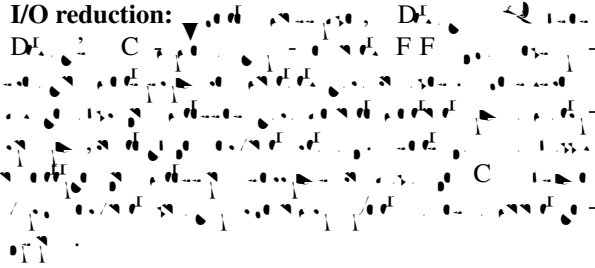
D#

```

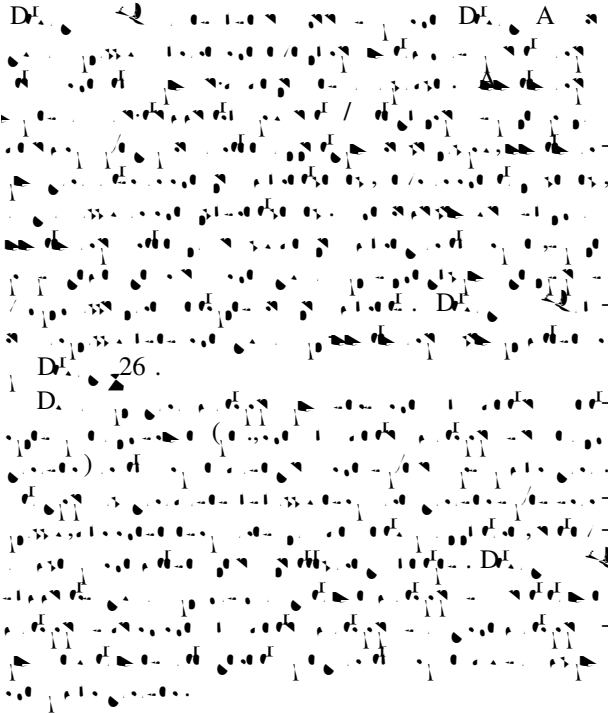
Eager Aggregation:

```


```



4.2.2 Dynamic Optimizations



4.2.3 Optimizations for OrderBy

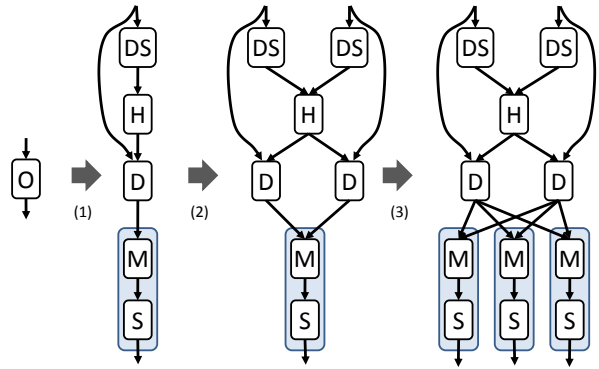
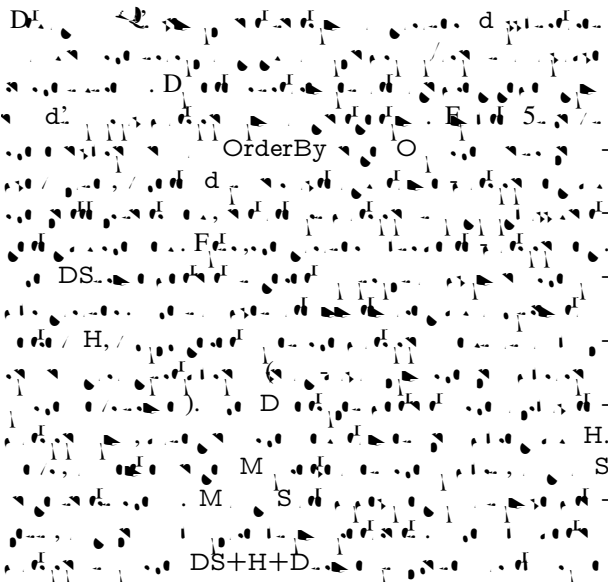


Figure 5: Distributed sort optimization described in Section 4.2.3. Transformation (1) is static, while (2) and (3) are dynamic.

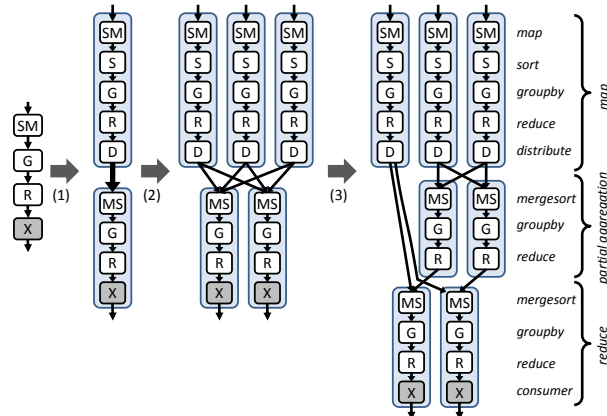
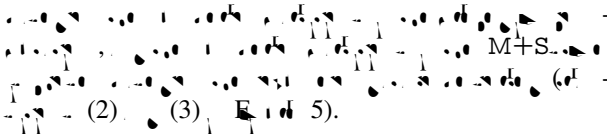
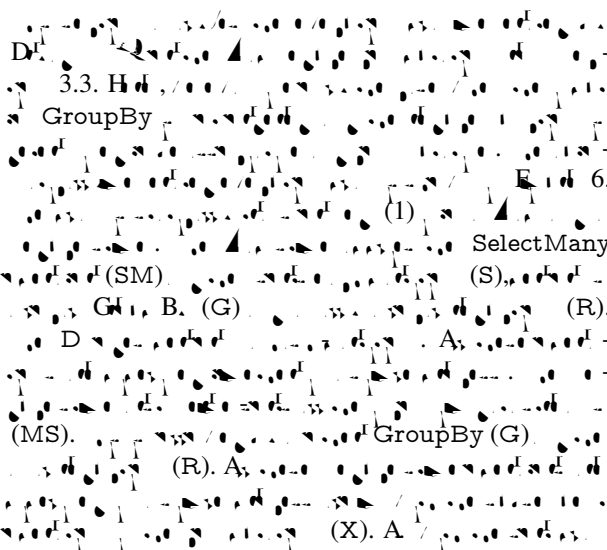


Figure 6: Execution plan for MapReduce, described in Section 4.2.4. Step (1) is static, (2) and (3) are dynamic based on the volume and location of the data in the inputs.



4.2.4 Execution Plan for MapReduce



4.2.3. ... (2)

... D₁ ...
 ... (3) ...
 ... G ...
 ... 15 ... D₁ ...
 ... 26 ... D₁ ...
 ... *all* ...

4.3 Code Generation

... E G ... D₁ ... E G ...
 ... D₁ ... D₁ ... E ...
 ... F ...
 (1) ...
 (2) ... E ...

... E G ... D₁ ... E G ...
 (1) ... F ...
 (2) ... E ... E ...

4.4 Leveraging Other LINQ Providers

... D₁ ... D₁ ...
 ... 19 ...
 ... D₁ ...
 ... 25 ...
 ... B ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... F ... D₁ ...

4.5 Debugging

... D₁ ... D₁ ...
 ... D₁ ...
 ... D₁ ...
 ... A ... 4.4, D₁ ...
 ... E ...

A D
 D
 D
 D

5 Experimental Evaluation

D
 A
 5.1. D
 D

5.1 Hardware Configuration

240 E
 2003 64
 A D 2218 HE C
 2.6 GHz, 16 GB, DD 2
 750 GB, A A
 2048
 48 GB, E GB
 E 29 31
 E
 2048
 6 802.3

6 GB
 \$1000

5.2 Terasort

D
 3 10
 100-B
 10-B
 3 D
 DryadTable
 OrderBy;
 4.2.3 (

3.87 GB
 (4,166,666,600 B.)
 n
 $3.87n$ GB
 $n = 240$
 10^{12} B.
 H
 D
 10^{12} B., 150 GB

1 240,
 3.87 GB, 10^{12} B.,
 5%
 E 7
 D
 F $2 \leq n \leq 20$
 $n > 20$

G	1	2	10	20	40	80	240
	119	241	242	245	271	294	319

1:
 n -
 $3.87n$ GB,
 10^{12} B., $n = 240$.

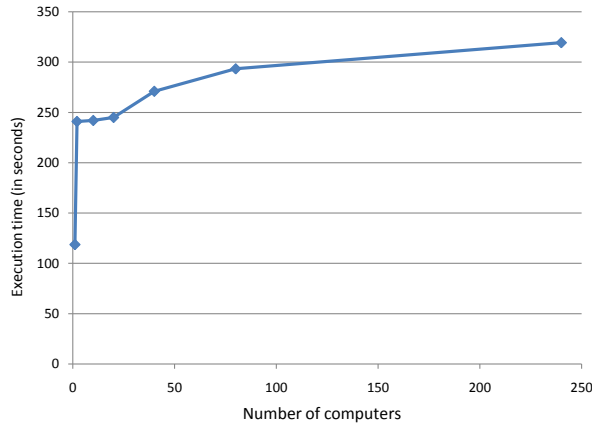


Figure 7: Sorting increasing amounts of data while keeping the volume of data per computer fixed. The total data sorted by an n -machine experiment is around $3.87n$ GBytes, or 10^{12} Bytes when $n = 240$.

Group	1	5	10	20	40
Dryad	2167	451	242	135	92
DryadLINQ	2666	580	328	176	113

Figure 8: Speed-up of the Skyserver Q18 computation as the number of computers is varied. The baseline is relative to DryadLINQ job running on a single computer and times are given in Table 2.

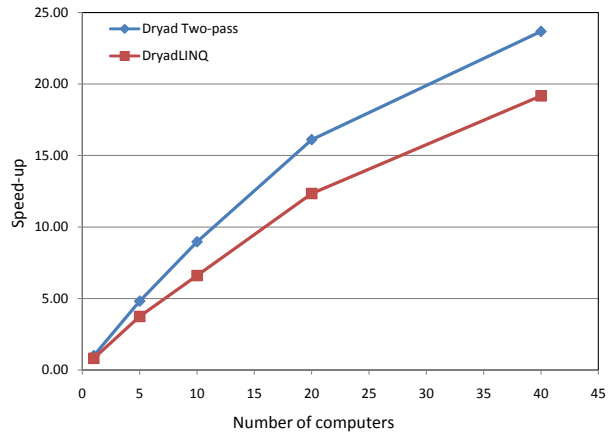
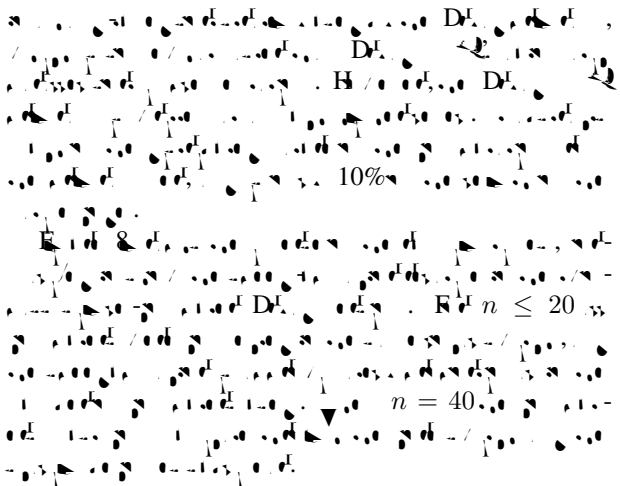


Figure 8: The speed-up of the Skyserver Q18 computation as the number of computers is varied. The baseline is relative to DryadLINQ job running on a single computer and times are given in Table 2.

Figure 9: Performance comparison of Dryad and DryadLINQ for sorting data. The graph shows execution time vs number of computers, with DryadLINQ consistently performing better than Dryad.

5.3 SkyServer

Figure 10: Performance comparison of Dryad and DryadLINQ for SkyServer Q18. The graph shows execution time vs number of computers, with DryadLINQ consistently performing better than Dryad. The data points are: (1, 2167), (5, 451), (10, 242), (20, 135), (40, 92) for Dryad and (1, 2666), (5, 580), (10, 328), (20, 176), (40, 113) for DryadLINQ.



5.4 PageRank

Figure 12: Performance comparison of Dryad and DryadLINQ for PageRank. The graph shows execution time vs number of computers, with DryadLINQ consistently performing better than Dryad. The data points are: (1, 2167), (5, 451), (10, 242), (20, 135), (40, 92) for Dryad and (1, 2666), (5, 580), (10, 328), (20, 176), (40, 113) for DryadLINQ.

93, 35,

H.

80%-90%

()

H

(+7 C),

(+21 C),

(+18 C).

38.

(f

240

954 16.5B, 1.2 B

10 12,792

116GB, 10

690.

H

B, D

5.5 Large-Scale Machine Learning

D

2.1GB.

3, 3.3

160, C# (1)

(2) (3)

E

10

5

5 C

(), /

10,000

240

11, 5

C

D

33

(514).

F

11

(180GB,

40GB).

D

(host, hour)

E

(95%

174,588

4, 22

10 C

6 Related Work

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

D

...), ... 18, ... G ... 17,
B ... 8, ... 22, ...
DB2 ... E ... 4, ... 20.

A, ... D,
... F ... D,
(DD) ... (DA)

DD
D ... H
4.2 ... D

... D
... (...)
... D

E
... D
... A
... D
... D

... 16.

6.2 Large Scale Data-Parallel Computation Infrastructure

... C ...)
... 13, ...

G ... 15
H ... 5
D ...

...

A ... G
B ... 11, A ... DB,
... D ... A ... D
... D ...

6.3 Declarative Programming Languages

... H ... 37, C ... 7,
E ... 6.

... R ...
... 32
... 31 ... H ...
... 12 ... 9,
H E (... F ... H ...)
... 10, ... 26, ... (...)
... D ...)
G ... (...)
... E ...
... 39
... 36 ... D ...
... 30,

7 Discussion and Conclusions

D ...
... R ...
... D ...

D ...
D ...
D ...

D ...
E G ... D ...
D ...

D ...

D₁ D₂ D₃ D₄ D₅ D₆ D₇ D₈ D₉ D₁₀ D₁₁ D₁₂ D₁₃ D₁₄ D₁₅ D₁₆ D₁₇ D₁₈ D₁₉ D₂₀ D₂₁ D₂₂ D₂₃ D₂₄ D₂₅ D₂₆ D₂₇ D₂₈ D₂₉ D₃₀ D₃₁ D₃₂ D₃₃ D₃₄ D₃₅ D₃₆ D₃₇ D₃₈ D₃₉ D₄₀ D₄₁ D₄₂ D₄₃ D₄₄ D₄₅ D₄₆ D₄₇ D₄₈ D₄₉ D₅₀ D₅₁ D₅₂ D₅₃ D₅₄ D₅₅ D₅₆ D₅₇ D₅₈ D₅₉ D₆₀ D₆₁ D₆₂ D₆₃ D₆₄ D₆₅ D₆₆ D₆₇ D₆₈ D₆₉ D₇₀ D₇₁ D₇₂ D₇₃ D₇₄ D₇₅ D₇₆ D₇₇ D₇₈ D₇₉ D₈₀ D₈₁ D₈₂ D₈₃ D₈₄ D₈₅ D₈₆ D₈₇ D₈₈ D₈₉ D₉₀ D₉₁ D₉₂ D₉₃ D₉₄ D₉₅ D₉₆ D₉₇ D₉₈ D₉₉ D₁₀₀

D₁ D₂ D₃ D₄ D₅ D₆ D₇ D₈ D₉ D₁₀ D₁₁ D₁₂ D₁₃ D₁₄ D₁₅ D₁₆ D₁₇ D₁₈ D₁₉ D₂₀ D₂₁ D₂₂ D₂₃ D₂₄ D₂₅ D₂₆ D₂₇ D₂₈ D₂₉ D₃₀ D₃₁ D₃₂ D₃₃ D₃₄ D₃₅ D₃₆ D₃₇ D₃₈ D₃₉ D₄₀ D₄₁ D₄₂ D₄₃ D₄₄ D₄₅ D₄₆ D₄₇ D₄₈ D₄₉ D₅₀ D₅₁ D₅₂ D₅₃ D₅₄ D₅₅ D₅₆ D₅₇ D₅₈ D₅₉ D₆₀ D₆₁ D₆₂ D₆₃ D₆₄ D₆₅ D₆₆ D₆₇ D₆₈ D₆₉ D₇₀ D₇₁ D₇₂ D₇₃ D₇₄ D₇₅ D₇₆ D₇₇ D₇₈ D₇₉ D₈₀ D₈₁ D₈₂ D₈₃ D₈₄ D₈₅ D₈₆ D₈₇ D₈₈ D₈₉ D₉₀ D₉₁ D₉₂ D₉₃ D₉₄ D₉₅ D₉₆ D₉₇ D₉₈ D₉₉ D₁₀₀

Acknowledgements

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

References

1. BA, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
2. BA, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
3. BA, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
4. BA, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

- 5 G. DB2. IBM Systems Journal 34, 2, 1995.
- 6 BECKER, D., GAARD, O., AND ADAR, O. International Parallel and Distributed Processing Symposium (IPDPS), 2005.
- 7 BECH, G. E. Communications of the ACM (CACM) 39, 3, 1996.
- 8 BIRNEY, D., O'EG, C. F., AND BIRNEY, C. E., AND ADAR, H., AND H. C. ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), 1995.
- 9 BIRNEY, H., AND ADAR, C. A., AND CEAD, G., AND F. H., AND F. A., AND B., AND H., AND ADAR, D. E. IEEE Trans. on Knowl. and Data Eng. 2, 1, 1990.
- 10 CAHILL, O., AND DEC, E. D. W4: Learning in Web Search, 2005.
- 11 CHAI, E., AND O'EG, B., AND ADAR, B., AND H. B., AND ADAR, H., AND O'EG, C. E. International Conference of Very Large Data Bases (VLDB), 2008.
- 12 CHAI, F., DEAN, O., GHEA, A., AND H. E., AND C., AND ACH, D. A., AND CHAI, D. A., AND F. E., AND ADAR, G. B. Symposium on Operating System Design and Implementation (OSDI), 2006.
- 13 CHAI, H., AND DA, A., AND H. A., AND ADAR, E., AND D. SIGMOD international conference on Management of data, 2007.
- 14 CHAI, A. G., H., AND AG, AND ADAR, H. Symposium on Principles and practice of parallel programming (PPoPP), 2003.
- 15 CAHILL, O., DAGE, E. B., AND DGH, H. B. ACM SIGMOD, 2004.
- 16 DEAN, O., AND GHEA, A., AND H. E., AND C., AND ACH, D. A., AND CHAI, D. A., AND F. E., AND ADAR, G. B. Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI), 2004.
- 17 DEHADE, A., AND ADAR, A., AND H. A., AND ADAR, E., AND D. Foundations and Trends in Databases 1, 1, 2007.
- 18 DEAN, D., AND GHA, DEHA, AND DEH, AND CH, E. DE, D., AND H. A., AND B. C. E., AND ADAR, E., AND G. IEEE Transactions on Knowledge and Data Engineering 2, 1, 1990.
- 19 DEAN, D., AND DGA, O., AND H. A., AND ADAR, E., AND D. Communications of the ACM 36, 6, 1992.
- 20 DEAN, O., AND ADAR, O., AND H. A., AND ADAR, E., AND D. Proceedings of the 2007 workshop on Declarative aspects of multicore programming, 2007.
- 21 EGEE, G. A., AND E., AND ADAR, A., AND H. A., AND ADAR, E., AND D. Sigmod Record, 1995.
- 22 FEIG, H., AND A., AND C., AND ADAR, G., AND H. B. International Conference on Extending Database Technology, 1998, H.-O., F., AND G. A., AND E., 1377. Lecture Notes in Computer Science.
- 23 GAARD, O., AND GAARD, A., AND HA, A., AND ADAR, E., AND D. GH, C., AND ADAR, D., AND ADAR, B. E. G., AND D. Distributed Data and Structures 4: Records of the 4th International Meeting, 2002.
- 24 HAARD, F. E. C., D., AND ADAR, E., AND D. SIGMOD Rec. 25, 3, 1996.
- 25 HE, E., AND O'EG, B. A., AND ADAR, H., AND O'EG, C. E. Foundations and Trends in Databases 1, 2, 2007.
- 26 ADAR, B. D., AND B. E., AND A., AND ADAR, E., AND D. D. Proceedings of European Conference on Computer Systems (EuroSys), 2007.
- 27 ABAR, AND ADAR, D. O. E. SIGMOD International Conference on Management of Data, 1998.
- 28 ADAR, D. ACM Comput. Surv. 32, 4, 2000.
- 29 ADAR, F., AND ADAR, E., AND A. D. Symposium on Applied computing (SAC), 2002.
- 30 GEE, D., AND ADD, AND FE, O., AND G. E., AND G. B. E., AND C., AND H., AND D., AND A., AND D., AND C. C., AND E. GE, AND A., AND ADAR, E., AND C. Concurrency and Computation: Practice and Experience 18, 10, 2005.
- 31 EED, B., AND A. A., AND ADAR, AND A. International Conference on Management of Data (Industrial Track) (SIGMOD), 2008.
- 32 DE, AND D. ADAR, AND G. E. E., AND ADAR, AND A., AND D. Scientific Programming 13, 4, 2005.
- 33 ADAR, A., AND D. ADAR, AND ACC, AND C., AND BA, HA, AND B. AC, AND AAC, AND ADAR, E., AND C. International Conference on Uncertainty in Artificial Intelligence, 2008.
- 34 EB, A., AND BEA, C., AND E. E. E., AND CHE, AND AC, AND GE, AND HACHE, AND HA, AND FE, O., AND GE, O., AND ADAR, AND B. Conference on Innovative Data Systems Research (CIDR), 2005.
- 35 EB, A., AND ADAR, AND ABAD, D. O., AND HA, AND HACHE, AND ADAR, AND H. A., AND D. International Conference of Very Large Data Bases (VLDB), 2007.
- 36 ADAR, AND H. E. D., AND A. G., AND ADAR, AND A. Workflows for e-Science. 2007, 320-339.
- 37 DE, AND D. H., AND ADAR, AND H. Journal of Functional Programming 12, (4&5), 2002.
- 38 ADAR, AND FE, E., AND D., AND B. D., AND E. G., AND G. DA, AND C. E., AND O., AND CHE, AND F., AND ADAR, AND D. 2008-74, 2008.
- 39 HA, AND HA, E. G., AND C. FF, D., AND B., AND F. E., AND A., AND E., AND G., AND EF, E. A., AND AC, AND EF, A., AND ADAR, DE, AND F. IEEE Congress on Services, 2007.