shows the base and the potential, a database application will benefit from the KSR1 as a scalable multiprocessor computer.

Further work will include the tuning of queries, the investigation of Oracle Parallel Server on the KSR1 and we will report scalability figures for more than 32 processors.

Acknowlegdements

This work was done in a joint project of Siemens Nixdorf Informationssysteme AG (SNI), the Fachhochschule Nordostniedersachsen, Lüneburg and the Computing Center of the University of Mannheim.

We like to thank Kendall Square Research, Waltham and especially David Wheat for their assistance and fruitful discussions.

The funding of this work was partly done by SNI.

References

[1] *Oracle7 Parallel Query Option*, Performance Report, Oracle, 1993

[2] Schumacher, R. (Ed.): *One Year KSR1 at the University of Mannheim*, University of Mannheim, RUM 35/93, December 1993.

[3] Kendall Square Research Corporation: *Technical Summary,* Waltham, MA (USA) 1992.

[4] Tseng, E. and Reiner, D.: *Parallel Database Processing on the KSR1 Computer*, ACM SIGMOD International Conference on Management Data, 1993

[5] Reiner, D., Miller, J. and Wheat, D.: *The Kendall Square Query Decomposer*, Proc. of *Second Parallel and Distributed Information Systems Conference*, San Diego (USA), 1993.

[6] Kendall Square Research Corporation: *Oracle for the KSR Computer,* Waltham, MA (USA) 1994.

For all queries we see the following results on 16 processors (Fig 4). On an average the efficiency is about 50%, with a spread from 24% to 93%, with the latter being nearly linear speedup. The total runtime for a query on one processor was in the range from several minutes up to 3 hours, depending on the query. Having in mind that none of the queries were tuned, nor by hashing or clustering neither by hints to the Oracle optimizer, this result was very impressive for us.

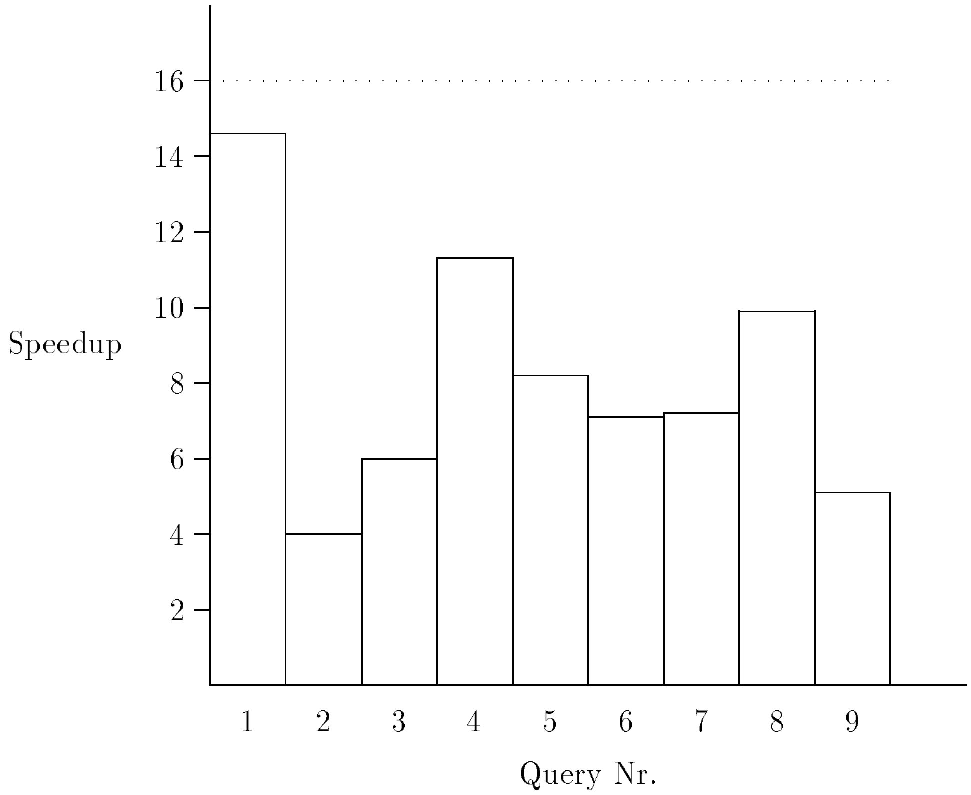

Figure 4: Speedup for different queries

5 Conclusions

In a first project we evaluated Oracle with the Query Decomposer on the KSR1. The parallelization of typical DSS–type queries was done in 7 out of 9 cases automatically by QD. The other 2 queries could be reformulated easily to suit the needs of the decomposer. The software is stable, and the user has the full capabilities of Oracle. Explicitly we found, that the coarse grain parallelism of DSS applications fit well for parallelization. The KSR1 with its shared memory concept provides a hardware platform, which supports the type of parallelism underlying database applications.

An average of 50% efficiency for untuned queries and a peak of over 90% efficiency

Query	Description
7	Join 3 tables, 3 predicates, 1 aggregate, 1 group, 1 order Changes: Removed 5 columns from group by to reduce the output volume. The volume of data searched was not changed.
8	Join 2 tables, 3 predicates, 1 aggregate, 1 order, 2 decodes Changes: Rewrote to eliminate UNION ALL
9	Join 2 tables, 2 predicates, 1 aggregate, 1 group, 1 order Changes: Eliminated the view

Also indicated are the changes we had to do. Only to query No. 6 and 8 we had to make substantial but obvious changes. However the results were of course the same. All other changes were done for convenience. We did not optimize the queries by introducing clustering or hashing of tables and indexes.

4.2 Results

As mentioned, the tables have to be partitioned into files, not necessarily spread over several disk, however this might lead to I/O contention. To see the influence of the possible contention of different processes making their I/O to the same disk we spread the biggest table (1 GB) out on 16 files and distributed these on 4 or 16 disk drives. For 16 processes we see an increase in the speedup of 13.4 to 14.9 which means an increase in the efficiency of about 10% from 84% to 93% (Fig 3). The measurements were done with query 1.

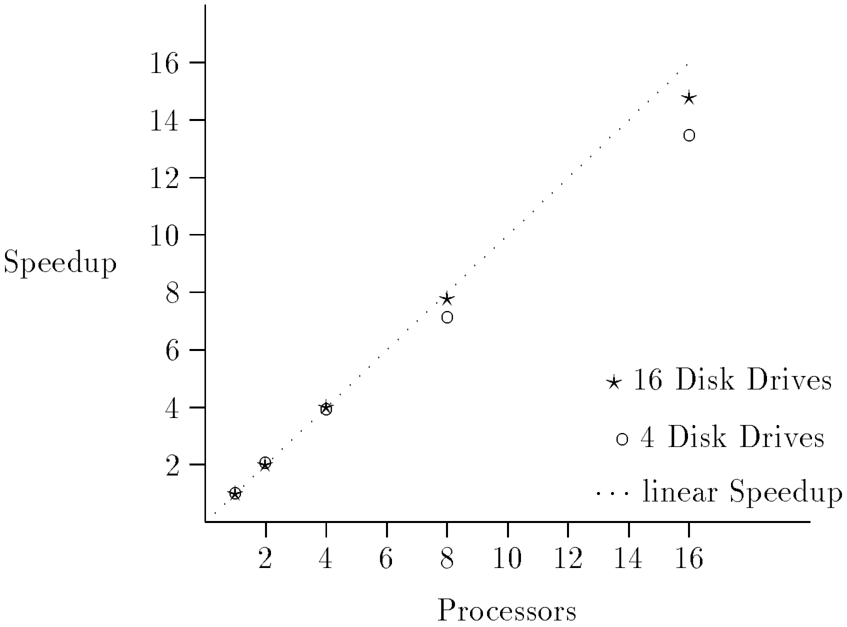

Figure 3: Speedup for a parallel 'full scan' query on a 1 GB db-table distributed on 4 or 16 disk drives

```
4 3 TABLE ACCESS FULL EMP
5 3 TABLE ACCESS BY ROWID DEPT
  6 5 INDEX UNIQUE SCAN PK_DEPT
```

The Oracle optimizer has table 'EMP' selected to be the driving table. Because this table is partitioned, a subquery will be executed on each partition.

The user has the possibility via UNIX environment variables and/or on per-query basis via directives to influence QD–behaviour, e.g. enable or disable QD. Also SQL–scripts available which after invocation gather statistics on I/O, cputime etc.

4 The Database Setup and Results

Typical sizes of databases, for which the performance of MPP are necessary to perform decision support queries are in the range of 50 GB as a minimum. As a starting point we have choosen a database with five tables using about 5 GB disk space, the biggest table with about 6 000 000 Rows. This database was choosen as it provides a typical setup for decision support applications. It can easily be extended to a realistic size, which we might do in the future.

4.1 The Queries

The queries used were choosen to represent typical DSS requirements. As we can not disclose the text of the queries, the following table describes the functionality of each query.

Query	Description
1	Full table scan, 1 predicate, 8 aggregates, 2 group, 2 order Changes: none
2	Join 3 tables, correlated subquery, 3 predicates, 1 aggregate Changes: none
3	Join 3 tables, 3 predicates, 1 aggregate, 2 group, 2 order Changes: none
4	Full table scan, 4 predicates, 1 aggregate Changes: none
5	Join 4 tables, 6 predicates, 1 aggregate, 3 group, 3 order Changes: Eliminate the view
6	Join 5 tables, 4 predicates, 2 aggregates, 1 group Changes: Eliminated all 4 views

Queries with an `union all` statement, and queries referencing more than one view can not be handled by QD and will be executed serially.

The parallelization is done by splitting the query into subqueries, each executed in an own subprocess. After processing the results of the subprocesses joined by QD, depending on the query. The essential prerequisite for parallelization of SQL queries on the KSR is, that the major tables are spread out on different files or raw-devices. The QD chooses as a starting point for parallelization the so-called 'driving table' selected by Oracle (in most cases the table with the highest number of rows). The number of files on which this table is spread over limits the maximal parallelization (Figure 2).

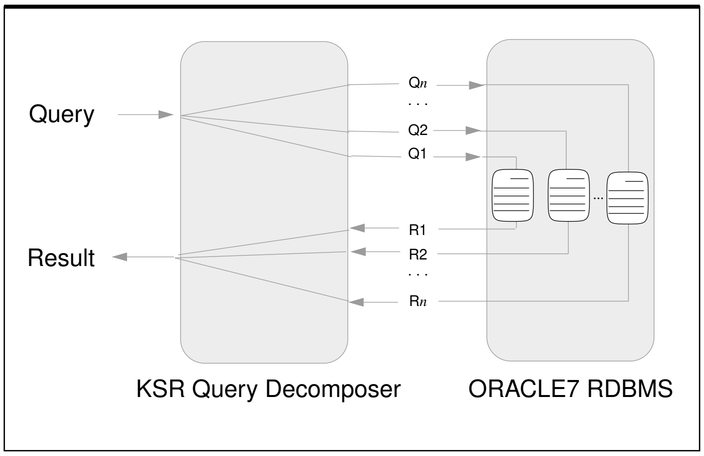

Figure 2: The Query Decomposer passes subqueries to Oracle and joins the results

KSR also extended Oracle's explain facility to show whether the query will be executed in parallel, and which table was choosen as the driving table. This is best seen in the following example:

```
EXPLAIN PLAN FOR

SELECT DNAME, AVG(SALARY) FROM EMP, DEPT
  WHERE EMP.DNO = DEPT.DNO
  GROUP BY DNAME;

EXPLAINED

QUERY_PLAN
--------------------------------------------
1 0 KSR PARALLEL EXECUTION AGGREGATION EMP
  2 1 SORT GROUP BY
    3 2 NESTED LOOPS
```

3.2 The Query Decomposer

The Query Decomposer (QD) was developed by KSR, to parallelize SQL queries. The QD is built into the user programming interface (UPI) [5], which is Oracles connection to the outside world (Figure 1). After passing the query to Oracle,

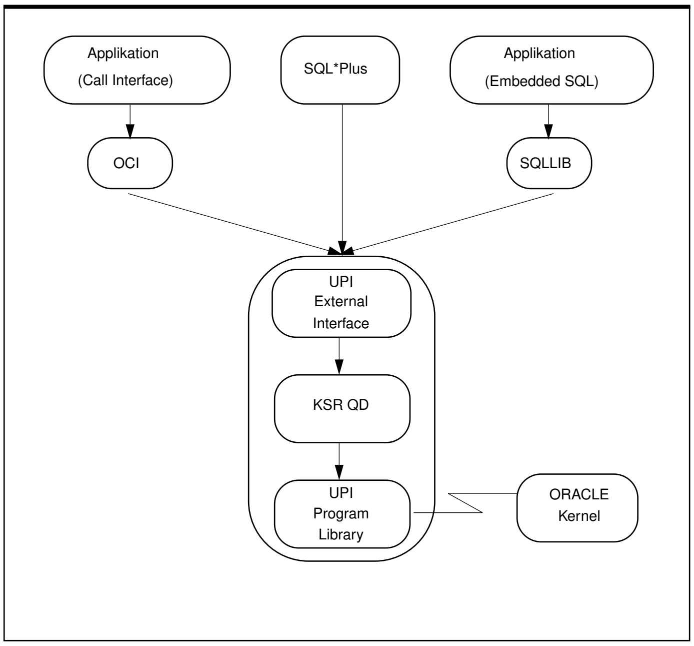

Figure 1: Location of the Query Decomposer

either by SQL*plus or via embedded SQL or even with SQL*net from a remote machine, the QD decides whether the query will be parallelized. In the simple case, that no possibility for parallelization is found, the query is passed without modification to Oracle. So the process of parallelization is fully transparent for the user. The following SQL statements or constructs are handled by QD [6]:

- joins, i.e. equijoins, nonequijoins, outer joins, cartesian products
- all aggregates
- nested aggregates (e.g., `sum(count(*))`)
- subqueries and correlated subqueries
- `group by`, `order by`, `having` clauses
- `insert/select` clause

with the 'Query Decomposer' (a product of KSR) is explained. In the last section the database setup and performance results are shown.

2 KSR1

The KSR1–32 is the central compute server of the Computing Center of the University of Mannheim. It consists of 32 processors each with 32 MB local cache (DRAM memory-chips, operated as cache) adding up to a total of 1 GB main memory. The proprietary superscalar 64 bit RISC processor is running at 20 MHz and executing two instructions each cycle. This yields a peak performance of 40 Mflop/s or 40 Mips for a single processor [3]. The system is equipped with 20 GB local diskspace. For further user data a disk-array is mounted via NFS. The connection to the outside world is realized by Ethernet within the computing center and FDDI, when coming from the institutes of the university or the internet.

The KSR1 runs the OSF/1 operating system symmetrically on all processors. You can partition the KSR1 into processor sets dynamically by software. At present we use three different processor sets, the default set with 8 processors is mainly for interactive tasks, the small set with 4 processors for testing purposes, and the big set with 20 processors for production. All tests and measurements were made in multi-user mode of the machine using the big set of processors.

3 Oracle on the KSR1

3.1 Installation of Oracle on the KSR1

All tests were done with ORACLE Version 7.0.13. The installation is simply done by reading an exabyte tape. You need to run only one instance of Oracle, this is different from most other distributed memory parallel computer, where you have one instance on each processor, however you can have more than one instance [4]. The system utilization of one instance is 4 processes and ca. 20 MB main memory. We did tests with 200 MB main memory, but got no significant speedups.

All maintenance is the same as for any other unix–workstation. Also under heavy load of the system we experienced no problems running the database software. As already mentioned, all testing – except for tests on 32 processors were done with other users on the machine, however they were not allowed to share the same set of processors we used.

Oracle on the KSR1 Parallel Computer

Heinz Kredel, Robert Schumacher, Erich Strohmaier
Computing Center, University of Mannheim, D–68131 Mannheim, Germany
e-mail:parallel@rz.uni-mannheim.de

RUM 39/94, August 1994

Abstract

Database applications for decision support (DSS) are often said to be among the most promising commercial applications for MPP systems. We installed Oracle on our KSR1 parallel computer. In a first project we tested the KSR Query Decomposer, a software, which automatically and transparently parallelizes SQL-queries. We applied Oracle with the Decomposer builtin to typical DSS-type queries and got preliminary, but very promising performance results and speedups of 93% on 16 processors.

1 Introduction

Database applications mostly fall in 2 categories: Online Transaction Processing (OLTP) or Decision Support Systems (DSS). Both types of operation are basically different from each other, OLTP–type statements are shortrunning, but for a typical application there are hundreds of thousands of these statements. Such a statement is a lookup or an update or an insert of a single row. DSS–type queries are longrunning, during the processing of such a query, multiple tables are joined, mostly one of these tables has to be scanned as a whole. Those queries are also called killer–queries, because of the load, they put on a system. Upcoming applications like the data–warehouse from where you can order almost everything out of your TV–chair, require both types of operations. OLTP–type statements will be involved with all the ordering and delivering, DSS–type queries will be issued to produce a customer profile.

They cpu–time requirements for DSS is already now not deliverable with the outfashioned mainframe. As a consequence the databases now are ported to parallel computer, from which are expected that they can deliver the power necessary to bring the turn–around time for DSS application down from days to minutes [1].

The Mannheim Computing Center acquired in December '92 a 32 processor KSR1 parallel computer, which is manufactured by Kendall Square Research (KSR). From the beginning we had a broad range of applications on the system [2]. In this report we present our experience with the KSR1 as a database server especially for DSS–type queries. After a short description of the KSR1 hardware, the implementation of Oracle V 7.0 on the KSR1 and the parallelization of queries