Solving the Multi-operator Placement Problem in Large-Scale Operator Networks

Stamatia Rizou, Frank Dürr, Kurt Rothermel
Universität Stuttgart, Institute of Parallel and Distributed Systems, Universitätsstraße 38, 70569 Stuttgart, Germany
{stamatia.rizou|frank.duerr|kurt.rothermel}@ipvs.uni-stuttgart.de

***Abstract*—Processing streams of data in an overlay network of operators distributed over a wide-area network is a common idea shared by different applications such as distributed event correlation systems and large-scale sensor networks. In order to utilize network resources efficiently and allow for the parallel deployment of a large number of large-scale operator networks, suitable placement algorithms are vital that place operators on physical nodes.**

In this paper, we present a distributed placement algorithm that minimizes the bandwidth-delay product of data streams between operators of the network in order to reduce the induced network load. Since the fundamental optimization problem is NP-hard, we propose a heuristic solution. First, we calculate an optimal solution in an intermediate continuous search space, called latency space. Subsequently the continuous solution is mapped to the physical network. Our evaluations show that this algorithm reduces the resulting network load significantly compared to state of the art algorithms and achieves results close to the optimum.

I. Introduction

Operator networks are a class of overlay networks that are used for distributed in-network processing of data streams. In these networks, data originating at a set of distributed data sources is jointly processed by a set of linked operators implementing data processing functionalities, such as filtering or aggregation. Operator networks have a number of important and popular applications including complex event processing (CEP), data stream processing, and global sensor networks.

Operator networks may host many stream processing tasks, which may involve a huge number of potentially widely dispersed data sources. Also the applications issuing stream processing tasks to the network may be widely distributed, and the data streams communicated between operators often consume a substantial amount of network bandwidth, such as a video stream captured by a camera. For those large-scale operator networks the consumption of network resources is a critical issue as it has a strong impact on the scalability of the system. Therefore, when mapping operators to the available hosts we should try to find a placement that minimizes the consumption of network resources. This problem is tackled by so-called *operator placement algorithms* [3], [8].

The main contribution of this paper is a novel distributed operator placement algorithm which minimizes the network usage of stream processing in operator networks. Our algorithm solves the so-called Multi-operator Placement (MOP) problem, which seeks for an operator placement minimizing the aggregated bandwidth-delay product of all streams in the operator network. The bandwidth-delay product is a measure for the load put onto underlay network paths (links and routers). Obviously, reducing network usage contributes to the scalability of the operator network.

More precisely the MOP problem is equivalent to the Multi-facility Location Problem from operations research [11]. Similar to the Multi-facility Location Problem, the MOP problem is NP-hard if solved for a *discrete* set of (physical) nodes available for hosting operators. Therefore, we first solve the *continuous* Multi-operator Placement problem, where we assume a virtual node to exist at each position in the so-called latency space [6], which is used to model delays between nodes in the underlay network. Together with a formal proof of correctness, we propose a distributed algorithm, where each operator autonomously finds an optimal virtual node in the latency space. In a second step, the selected virtual nodes are mapped to the available physical nodes in the latency space. According to our general goal to use communication resources efficiently, we optimized the distributed operator placement algorithm for low communication overhead by reducing the number of management messages and operator migrations.

Although we will show that our algorithm finds an optimal solution for the continuous MOP problem, the final mapping of virtual nodes to physical nodes may lead to a suboptimal solution of the original NP-hard problem. However, our evaluations show that with our heuristic approach the network usage is only 14% greater on average than the optimal solution. Furthermore, our algorithm outperforms significantly the currently best state-of-the-art operator placement algorithm optimizing network usage (SBON [8]), both in terms of the network usage of data streams and network overhead.

The rest of this paper is structured as follows. In Section II,we discuss the related work. In Section III, we introduce our system model. Based on the system model we formally state our problem in Section IV, in Section V we present a distributed algorithm for solving the Multi-operator placement problem. In Section VI we show in detail how the solution of the MOP problem is computed through local optimization. In Section VII we provide a formal proof of correctness for our distributed algorithm. Finally in Section VIII we evaluate the performance of our algorithm by comparing it to the theoretic optimum and related approaches, before we conclude the paper in Section IX.

Published in International Conference on Computer Communication Networks
@IEEE 2010
DOI 10.1109/ICCCN.2010.5560127

II. Related Work

Lakshmanan et al. [7] provide a comprehensive overview of existing placement algorithms. Their study show that the diversity of optimization goals leads to different placement algorithms. Therefore, in the following discussion we will confine ourselves to placement algorithms that aim at optimizing network usage, since this is our optimization goal.

Ahmad et al. [3] at first proposed an approach for operator placement optimizing the bandwidth-delay product. With this approach, nodes are chosen that lie on the paths between two end-points of a DHT-based overlay network. However, in [9] Pietzuch et al. showed that looking for candidate nodes on DHT paths leads to a poor approximation of the optimal solution since the actual goal of the DHT routing tables is to minimize the number of hops rather than the network usage.

Closest to our approach is the work of Pietzuch et al. [8], who were the first to propose the usage of the latency space as an intermediate continuous search space. In their approach, called SBON, the operator placement in the latency space is based on a physical model of springs. The goal of the proposed algorithm is to minimize the overall energy of the corresponding physical system. However, in this model energy is proportional to the square of the latency while the network usage is only linear dependent on the latency. In other words SBON optimizes the metric $\mathrm{bandwidth} \times \mathrm{delay}^2$, which does not intuitively model network usage. For instance by doubling the length of a physical path between two operators, the number of bits in transit on this path is only doubled rather than quadrupled. In contrast, our algorithm actually optimizes $\mathrm{bandwidth} \times \mathrm{delay}$. Moreover, our algorithm fully exploits the locality of the problem by finding at each iteration the current local optimal solution, while SBON uses another model, which gradually moves at each iteration towards the local optimum. The evaluation shows that our algorithm outperforms SBON not only in the quality of optimization results, but also in terms of the communication and operator migration overhead induced by the placement algorithm.

III. System Model

We consider a system consisting of a set of (physical) nodes V, which are capable of hosting operators needed for stream processing. These nodes are connected through a communication network, such as the Internet, allowing nodes to communicate with each other. Similar to [8], we assume a so-called *latency space*, which is an n-dimensional Cartesian space, where every node ν has a position $\vec{x}_\nu \in V$ such that the Cartesian distance $d(\overline{\nu_i \nu_j}) = |\vec{x}_{\nu_i} - \vec{x}_{\nu_j}|$ between any pair of nodes ν_i, ν_j corresponds to the communication delay between these nodes. The latency space can be constructed efficiently in a distributed manner using delay measurements between physical nodes and an algorithm for calculating network coordinates such as [6]. To determine its position in the latency space, every node performs these calculations and provides this information to other nodes as described later. The set of coordinates of the physical nodes is denoted by $C(V)$.

In our execution model, a stream processing task is modelled as an *acyclic* directed graph of connected operators, called *operator graph.* The set of the deployed operator graphs constitutes the *operator network.* Operators, which perform any kind of stream processing operations, may have a number of incoming and outgoing streams. More precisely, a stream processing task is modelled as an operator graph $G = \{\Omega, S, A, L\}$ consisting of a set $\Omega = \{\omega_1, \ldots, \omega_n\}$ of *operators* that are connected by a set $L = \{\overline{\omega_1 \omega_i}, \ldots, \overline{\omega_j \omega_n}\}$ of *links.* A link is an unidirectional communication relationship between a pair of operators. Link $\overline{\omega_i \omega_j} \in \Omega \times \Omega$ connects operators ω_i and ω_j, where the former produces a stream that is communicated to and consumed by the latter. L_{ω_i} denotes the set of in- and out-going links attached to operator ω_i. Operators in set $S \subset \Omega$ only have outgoing links (producers of data streams) and hence are called *sources.* Set $A \subset \Omega$ denotes the set of *sink* operators, which only have incoming links and typically represent application entities. We assume that sinks and sources are pinned operators, i.e., their mapping to physical nodes is given and fixed, whereas every other operator can be freely assigned to any available node in V.

IV. Problem Statement

Our placement algorithm tries to minimize the network usage for each individual operator graph. Network usage is measured by the number of bytes that are in transit on the links of the operator graphs at a certain point in time. Formally speaking, the network usage of link $\overline{\omega_i \omega_j}$ is defined by the bandwidth-delay product $r(\overline{\omega_i \omega_j}) d(\overline{\omega_i \omega_j})$, where $r(\overline{\omega_i \omega_j})$ specifies the data rate of the stream communicated over that link and $d(\overline{\omega_i \omega_j})$ is the delay of that link. The link delay is defined by $d(\overline{\omega_i \omega_j}) = |\vec{x}_{\omega_i} - \vec{x}_{\omega_j}|$, where $\vec{x}_\omega$ denotes the position of the operator ω in the latency space.

Optimizing the placement according to this metric increases scalability as the communication load generated by operator graphs is minimized. Minimizing the network usage also helps to control network conquestion in the communication network. Note that the latency space is dynamic in the sense that the positions of physical nodes are continuously adapted depending on current delays. An overloaded path in the communication network leads to an increase of the delays between nodes using this path to communicate, and thus the distance between these nodes increases also. Since our placement algorithm dynamically adapts to changing node positions, the placement is adapted by choosing nodes that are close to each other, i.e., nodes whose communication paths are not overloaded. Finally, the optimization of the network usage also reduces the latency between operators in general as high latency paths are avoided.

To formalize our optimization problem, we first introduce the *Single-operator Placement (SOP) Problem*, which considers the optimal placement of a single unpinned operator, say ω, relative to the placement of its neighbours in the operator graph. Here, the optimization goal is to minimize the network usage of all the links connected to ω, i.e. the aggregated bandwidth-delay product of the links in L_ω is to be minimized. Equation 1 expresses the network usage associated with placement $\vec{x}_\omega$:

$$U_{\text{local}}(\vec{x}_\omega) = \sum_{\overline{\omega\omega_i} \in L_\omega} r(\overline{\omega\omega_i}) d(\overline{\omega\omega_i}) = \sum_{\overline{\omega\omega_i} \in L_\omega} r(\overline{\omega\omega_i}) |\vec{x}_\omega - \vec{x}_{\omega_i}| \tag{1}$$

The problem stated above can be extended to the so-called *Multi-operator Placement (MOP) Problem*, which seeks for the optimal placement of all unpinned operators of an operator graph $G = \{\Omega, S, A, L\}$, where the goal is to minimize the overall network usage of G. For a given placement $(\vec{x}_{\omega_1}, \ldots, \vec{x}_{\omega_n})$, G's network usage is defined as follows:

$$U_{\text{global}}(\vec{x}_{\omega_1}, \ldots, \vec{x}_{\omega_n}) = \sum_{\overline{\omega_i\omega_j} \in L} r(\overline{\omega_i\omega_j}) d(\overline{\omega_i\omega_j}) \tag{2}$$

This also minimizes the network usage for the entire operator network if all operator graphs in the operator network have only one sink, i.e. sinks do not share common operators.

Both SOP and MOP problems can be solved for *physical* and *virtual* nodes. In the latter case, we assume that there exists a virtual node at every possible position in the continuous latency space, i.e., $\vec{x}_\omega \in \Re^3$. Whereas in the former case, the operators can only be mapped to those positions in the latency space that are occupied by physical nodes, i.e. $\vec{x}_\omega \in C(V)$. Since the solution space can be either continuous $\Re^3$ or discrete $C(V)$, we distinguish between the continuous and discrete variant of the MOP and SOP problem.

V. Multi-operator Placement Algorithm

In this section, we propose a novel placement algorithm approximating the optimal solution of the discrete MOP problem. Due to the complexity of the discrete MOP problem, we propose here a heuristic approach that is based on the idea to solve first the correspoding *continuous* MOP problem and then map the selected virtual nodes to the available physical nodes to yield an approximation of the discrete MOP solution.

The continuous MOP problem can be solved in a distributed fashion by letting each unpinned operator autonomously solve the continuous SOP problem. In other words, each unpinned operator determines its optimal virtual node (i.e., its optimal position in the latency space) depending on the current virtual positions of its neighboring operators. We prove in Section VII that the collection of the continuous SOP solutions yields an optimal continuous MOP solution. The proposed distributed algorithm can be used for both the initial placement of an operator graph as well as for adapting the placement when delay or bandwidth conditions change significantly during the execution of an operator graph. For the initial placement, we simply execute the algorithm in a centralized way.

Algorithm 1 shows the algorithm performed by each unpinned operator ω. First, a solution for the continuous SOP Problem is calculated, i.e., the position of the virual node for ω is determined such that $U_{\text{local}}(\vec{x}_\omega)$ is minimal (line 1). For the algorithmic details of these calculations we refer to Section VI. In the next step, the selected virtual node is mapped to the closest available physical node ν (line 2) in the latency space. If ν_{new} differs from the current hosting node ν_{current}, the operator is migrated to the new physical node (lines 3–5), if the selected physical host is not overloaded after the deployment of the additional operator. Otherwise the algorithm excludes this physical node from the search space to prevent bottlencks and assigns the operator to the next nearest physical host. Typically, a number of iterations of the algorithm are required to approximate the optimal solution. To reduce the number of migrations a *lazy migration* strategy can be applied. In such a case, migrations can be delayed for some time without affecting the final outcome of the algorithm.

For mapping virtual nodes to the available physical nodes, we use a nearest neighbor search mechanism. As stated in Section III, we assume that the position of each physical node is known. Therefore, the nearest neighbor search can be realized using a distributed index as describe in [10]. For example, this index can be realized by the physical nodes forming a peer-to-peer network.

Algorithm 1 Multiple Operator Placement Algorithm

```
Require: $\omega$ is placed at physical node $\nu_{\text{current}}$
Require: Virtual coordinates of $\omega$'s neighboring operators
Require: Estimations of data rates of links in $L_\omega$
Ensure: $\omega$ is placed on optimal physical node
1: find $\vec{x}_\omega$ such that $U_{\text{local}}(\vec{x}_\omega)$ is minimal
2: find closest non-overloaded physical node $\nu_{\text{new}}$ with $\vec{x}_{\nu_{\text{new}}}$ to $\vec{x}_\omega$
3: if $\nu_{\text{new}} \neq \nu_{\text{current}}$ then
4:     migrate $\omega$ to $\nu_{\text{new}}$
5: end if
```

Finally the algorithm is executed in an event-driven manner. It is tiggered for operator ω in two cases: A neighbor operator informs ω that the neighbor's virtual node position or the data rate of a link connected to ω changes. The first case occurs whenever a neighboring operator calculates a new virtual node position when performing the algorithm. For detecting the second case, each unpinned operator measures the data rate of each incoming and outgoing link using an exponential moving average. For the initial placement we can estimate the data rate of each link according to the type of application or based on statistics gathered from previous deployments.

VI. Single-operator Placement Algorithm

Here we describe in detail the subalgorithm of Algorithm 1 (line 1) which finds the optimal SOP solution. The SOP problem corresponds to the well known Weber Problem [5], which asks for the position $\vec{x}_\omega$ that minimizes $U_{\text{local}}(\vec{x}_\omega)$ (Equ. 1). Since this function is known to be convex, approximation algorithms can be used to find the unique minimum of the function using an iterative algorithm.

Here we use a simple algorithm (Algorithm 2) that implements the gradient method for searching the minimum of $U_{\text{local}}(\vec{x}_\omega)$. Line 1 and 2 are the initialization steps that we discuss later. In each iteration, we first calculate

Algorithm 2 Single Operator Placement Algorithm

```
Require: Virtual node coordinates of $\omega$'s neighboring operators: $\{\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n}\}$
Require: Data rates of links in $L_\omega$: $\{r(\overline{\omega\omega_1}), \dots, r(\overline{\omega\omega_n})\}$
Ensure: $U(\vec{x}_\omega)$ is minimal
 1: $\vec{x}_\omega \leftarrow \text{Weber}_{\text{L}_1}\{r_1\vec{x}_{\omega_1}, \dots, r_n\vec{x}_{\omega_n}\}$
 2: $\vec{x}_\omega \leftarrow \text{CheckDeadPoints}\{\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n}\}$
 3: $step \leftarrow \max\{|\vec{x}_{\omega_1} - \vec{x}_\omega)|, \dots, |\vec{x}_{\omega_n} - \vec{x}_\omega)|\}$
 4: repeat
 5:     $\vec{f} \leftarrow \nabla U(\vec{x}_\omega) = \sum_i r(\overline{\omega\omega_i}) \times u(\vec{x}_{\omega_i} - \vec{x}_\omega)$
 6:     if $U(\vec{x}_\omega + step \times u(\vec{f})) < U(\vec{x}_\omega)$ then
 7:         $\delta \leftarrow U(\vec{x}_\omega + step \times u(\vec{f})) - U(\vec{x}_\omega)$
 8:         $\vec{x}_\omega \leftarrow \vec{x}_\omega + step \times u(\vec{f})$
 9:     else
10:         $step \leftarrow step/2$
11:     end if
12: until $\delta < \delta_t$
```

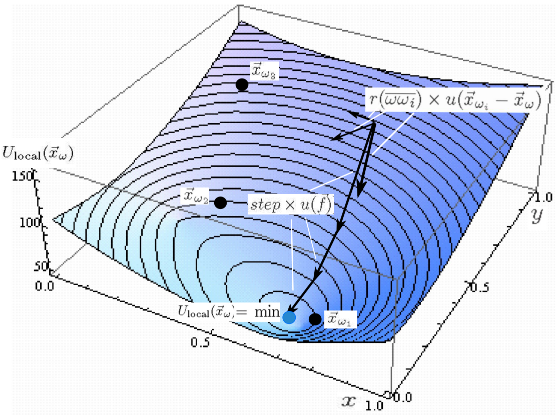

Fig. 1. Example of the gradient method for a 2-dimensional SOP problem.

the direction of the major flow $\vec{f}$, which corresponds to the gradient $\nabla U(\vec{x}_\omega)$ at the current position of operator ω (line 5). Then, we move towards this direction, which is given by the unit vector $u(\vec{f})$, with a certain step length *step* until we reach the minimum (cf. Figure 1). Initially we set the step to the maximum distance from ω to all of its neighbors (line 3) as the solution is restricted to the interior of the polygon that the neighbors form. If the current step length would overshoot the minimum, then it is halved (line 10). In each iteration the algorithm calculates the new network usage $U(\vec{x}_\omega)$ at the next estimated virtual position, and if this is smaller than the current network usage (line 6), it moves to the new virtual position (line 8) and sets as δ the absolute difference between the old and the new network usage (line 7). After a number of iterations the minimum is trapped and the algorithm terminates when the difference to the current network usage becomes smaller than a threshold δ_t (line 12).

The preprocessing steps help us to avoid some special cases, where the derivative is not defined and to improve the convergence speed of the algorithm. In detail, we first specify an approximation of the solution based on the solution of the corresponding Weber Problem for the Manhattan (L1) metric [4]. Then, we check for dead points (line 2), i.e. the points where the gradient is not defined. The candidate dead points for this problem are known to be the positions of neighbors. Therefore, if the network usage at one of the neighbors is lower or equal than the one given by the approximation, we use this point as initial position. Moreover in the special case where the solution coincides with the position of a neighboring operator, the two operators are clustered in one. Note that at the non-differentiable points the iterative method can escape by using a steepest descent direction based on the factors of all the neighbors for which the derivative can be defined.

VII. Proof of Correctness

If every operator independently optimizes its local position by solving the continuous SOP problem (Algorithm 2), then eventually *every* operator will be placed in a local optimal position. We call this solution an *all-local optimal solution*. Here we prove that an all-local optimal solution will also lead to a global optimal solution of the MOP problem.

First, we prove the following necessary condition:

Theorem 1: In the global optimal state, where $U_{\text{global}}(\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n})$is minimal, each operator ω is at its local optimal position such that $U_{\text{local}}(\vec{x}_\omega)$ is minimal.

Proof: We will prove this claim using a proof by contradiction: Assume there exists a minimal solution, $U_{\min} = U_{\text{global}}(\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n}) = \min$, such that there exists at least one operator ω_* that is not at its local optimal position, i.e., $U_{\text{local}}(\vec{x}_{\omega_*}) \neq \min$. (Otherwise there is nothing to prove since every operator is already at its local optimal position.) Then, the resulting global network usage is $U_{\min} = U_{\text{local}}(\vec{x}_{\omega_*}) + \sum_{\omega_j \in \Omega \setminus \omega_*} U_{\text{local}}(\vec{x}_{\omega_j})$.

Assume all operators besides ω_* are fixed to the places of the global minimal solution. Then, we can do a local optimization for ω_* by moving ω_* to a new position $\vec{x}'_{\omega_*}$ with $U_{\text{local}}(\vec{x}'_{\omega_*}) = \min$. The resulting global network usage is then defined as $U'_{\min} = U_{\text{local}}(\vec{x}'_{\omega_*}) + \sum_{\omega_j \in \Omega \setminus \omega_*} U_{\text{local}}(\vec{x}_{\omega_j})$.

Since $U_{\text{local}}(\vec{x}'_{\omega_*}) < U_{\text{local}}(\vec{x}_{\omega_*})$, $U'_{\min} < U_{\min}$, which is a contradiction to the assumption that $U_{\min}$ is minimal. ■

The sufficient condition that an all-local-optimal solution is always the optimal MOP solution, remains to be proven. To prove the sufficient condition we first show that *each* all-local optimal solution is a (possibly local) minimum of MOP:

Lemma 1: For any operator graph G, an all-local-optimal solution $(\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n})$ is also a local minimum of the global function $U_{\text{global}}(\vec{x}_{\omega_1}, \dots, \vec{x}_{\omega_n})$.

Proof: If the solution is at a differentiable point, the partial derivatives of U_{global} are equal to zero since it holds that $\frac{\partial U_{\text{global}}}{\partial \vec{x}_{\omega_j}} = \frac{\partial U_{\text{local}}}{\partial \vec{x}_{\omega_j}} = \sum_{\overline{\omega_j\omega_i} \in L_{\omega_j}} r(\overline{\omega_j\omega_i}) \times u(\vec{x}_{\omega_j} - \vec{x}_{\omega_i})$. Therefore the all-local optimal solution is a local minimum.

If one of the partial derivatives is not defined, then the solution of the corresponding SOP problem lies on a nondifferentiable point (dead point). According to our algorithm in this special case the operator will be clustered with its neighbor and the MOP solution is given by the solution of the clustered operator graph. Clustering is repeated until either it finds a differentiable solution for the clustered operator graph, which we have proved to be a local minimum, or it is degraded to a trivial SOP, whith one unpinned operator and a set of pinned neighbors (sources and sinks). Since SOP with only pinned neighbors is guaranteed to be solved optimally by our Single-operator Placement Algorithm, the all-local optimal solution is again a minimum of the global function.

Thus we have proven that an all-local optimal solution is also a minimum of the global function. ■

Finally we prove the sufficient condition that finalizes our proof of optimality.

Theorem 2: For any operator graph G, an all-local-optimal solution $(\vec{x}_{\omega_1}, \ldots, \vec{x}_{\omega_n})$ is also the unique minimum of the global function $U_{\text{global}}(\vec{x}_{\omega_1}, \ldots, \vec{x}_{\omega_n})$.

Proof: From Lemma 1 we know that an all-local optimal solution is a local minimum of U_{global}. Furthermore, we know that U_{global} has only one minimum since it is a convex function, i.e., if we have found a local minimum of U_{global} we also have found its global minimum. ■

VIII. Evaluation

A. Experimental Settings

To evaluate the performance of our algorithm we implemented our algorithm for the network simulator PeerSim.

For the underlying (physical) network topology, we used data gathered from a real network, namely the PlanetLab [2]. The *PlanetLab topology* consists of 226 physical nodes including real measurements of the delays between the nodes. The coordinates of the physical nodes in the latency space were found using a prototype implementation of the Vivaldi algorithm [1] that achieves to map the physical nodes in the latency space with an average error of 15 ms w.r.t. to the measured delays. The real PlanetLab topology gives us the chance to asses the practical performance of our algorithm in a real system.

Also the structures of the operator graphs to be deployed possibly influence the performance of the placement algorithm. Depending on the concrete experiment, we use *large operator graphs* of 12 nodes or *small operator graphs* of 6 nodes. Moreover, we assume that every operator has two or three children since we assume that this represents the usual case of an operator graph well.

The data rates on the links are generated randomly by varying the initial output data rates of the sources and the selectivity of the operators in a certain interval. The output data rates of the sources are distributed uniformly in the interval between 100 and 200 kbps. The selectivity of an operator is defined as the percentage of the output data rate with respect to the input data rate of the operator. Thus, operators with a selectivity close to 0 act as high selective filters in the network and generate very low output data rates, whereas operators with selectivity close to 1 generate output data rates equal to the incoming rate. In our evaluation, we vary the selectivity of the operators between 0 and 1.

We compare our placement algorithm, called *Multi-operator Placement Algorithm* (*MOPA*), to one state of the art algorithm called *SBON* [8]. As described in Section II, SBON also tries to optimize network usage using a different approach based on a spring relaxation model. Moreover, we compare our algorithm to the theoretical optimal placement algorithm (called *MOPopt*) solving the discrete MOP problem by using an exhaustive search.

B. Comparison of Discrete MOP Solutions

As a first step we investigated the quality of approximated discrete MOP solutions.

As performance metric, we use the *physical stretch factor*: $stretch_{\text{physical}} = \frac{\hat{U}_{\text{global,SBON|MOPA}}}{\hat{U}_{global,optimal}}$. $\hat{U}_{\text{global,SBON|MOPA}}$ denotes the network usage of a discrete MOP solution given by SBON and MOPA, respectively. $\hat{U}_{\text{global,optimal}}$ defines the optimal discrete MOP solution determined by MOPopt through an exhaustive search. This optimum serves as a reference of the approximated solutions achieved by SBON and MOPA. As the complexity of the discrete MOP problem increases exponentially with the number of nodes in the operator graph, we perform this evaluation only for small operator graphs.

Figure 2 shows the cumulative distribution of this experiment resulting from 1000 simulation runs. We see that in 70% of the measurements MOPA has a stretch factor lower than 1.1. The average stretch factor of MOPA is 1.14. Thus we see that although the number of physical nodes is small and the latency space sparsely populated with physical nodes, the optimal continuous MOP solution does not degenerate significantly after the physical mapping.

We also see that for 70% of the measurements, SBON has a stretch factor of 1.3 which is 16% higher than the physical stretch of MOPA for the same percentage. The average stretch factor of SBON is 1.29 compared to 1.14 for MOPA. Thus, MOPA keeps its theoretical advantage of having optimal continuous MOP solutions also after the mapping to physical nodes.

The highest stretch factor for SBON is 6.61, whereas the maximum of MOPA is only 3.67. In these cases the approximation is not close to the physical optimum. On the one hand, such a case can be caused by a bad mapping of a physical node in the latency space where the delays between physical nodes modelled in the latency space do not accurately reflect the real delays. On the other hand, the sparse character of the network topology can lead to bad discrete MOP approximations, where no well-matching physical node for the calculated virtual node position can be found.

C. Convergence

Finally, we evaluate the convergence properties of MOPA compared to SBON. We consider two metrics. First, we measure the induced *network overhead* denoted by the number of messages that have to be exchanged in order to communicate virtual node coordinates to neighboring operators whenever

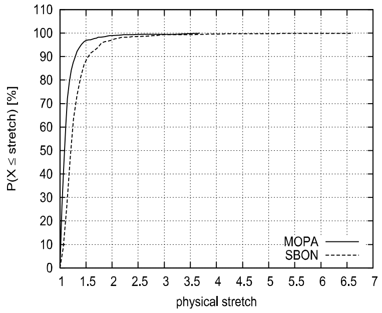

Fig. 2. Physical stretch factor of SBON and MOPA w.r.t. optimal discrete MOP solution.

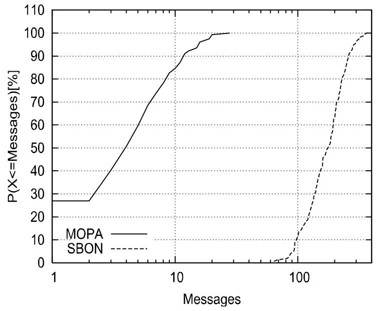

Fig. 3. Cummulative distribution of messages exchanged to reach a new solution.

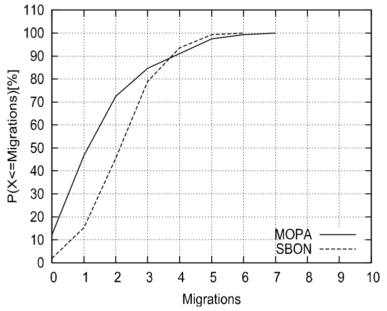

Fig. 4. Cummulative distribution of migrations to reach a new solution.

a new operator position has been calculated. Second, we measure the number of *operator migrations* that are performed until the equilibrium is reached. Since migrations largely outweigh local computations, the number of migrations is also an indicator for the convergence time.

For this experiment we use large operator graphs. We first let both algorithms converge to a stable solution. Then we generate a dynamic change by resetting the output data rates of all sources to new random values. The sudden change of the data rates provokes the re-placement of the operators. Note that a sudden change of *all* the data rates is a worst case scenario since the whole operator graph is affected. We placed 2000 operator graphs and measured the number of migrations and messages exchanged until the equilibrium is reached.

Figure 3 shows the cumulative distribution of the number of required messages for SBON and MOPA. We see that MOPA needs fewer messages to converge to a new equilibrium in all the cases. In detail, MOPA needs between 0 and 28 messages, while SBON needs 64 to 365 messages. On average MOPA only needs 3.25% of the messages that are needed by SBON. This superior performance of MOPA can be interpreted by two reasons. First our distributed algorithm moves in larger steps, while SBON moves slowly, making only small progress in each iteration. Second MOPA is more probable to create clusters, since the solution depends more on the data rates. In case of clustering, the operators lie on the same physical node and therefore the communication overhead is reduced.

Figure 4 depicts the cumulative distribution of the number of migrations on the physical network for MOPA and SBON. Again MOPA outperforms SBON by an average of 26.8% less migrations. Moreover we see that in 90% of the simulations MOPA needs less migrations than SBON. Similar to the number of exchanged messages, the reason for the smaller number of migrations of MOPA is the faster convergence due to large step size and clustering.

IX. Conclusions

In this paper, we formally introduced the Multi-operator Placement (MOP) Problem. Since this problem is NP-hard for a discrete set of physical nodes, we used a heuristic solution that first determines optimal operator positions for virtual nodes in a continuous search space and then maps the solution to the closest physical nodes to find an approximate solution of the discrete MOP problem. We presented a novel distributed algorithm that solves the continuous MOP problem and showed its correctness. Our evaluations showed that our algorithm finds solutions close to the optimum. Moreover it outperforms an existing placement algorithm in terms of quality of the found solutions as well as the communication overhead induced by the placement algorithm.

In future work, we are going to investigate the conditions of sharing operators among different operator graphs such that the network usage of the entire operator network is minimized. Furthermore, we are going to consider further optimization objectives such as end-to-end delay and load balancing, which are currently only considered indirectly.

References

[1] Network Coordinate Research at Harvard. http://www.eecs.harvard.edu/~syrah/nc/.

[2] Planetlab. http://www.planet-lab.org.

[3] Y. Ahmad and U. Çetintemel. Network-aware query processing for stream-based applications. In *VLDB '04*, pages 456–467, 2004.

[4] P. Bose, A. Maheshwari, and P. Morin. Fast Approximations for Sums of Distances, Clustering and the Fermat-Weber Problem. *Computational Geometry: Theory and Apllications*, 24:135–146, 2002.

[5] R. Chandrasekaran and A. Tamir. Algebraic optimization: the Fermat-Weber location problem. *Math. Program.*, 46(2):219–224, 1990.

[6] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *SIGCOMM '04*, 2004.

[7] G. T. Lakshmanan, Y. Li, and R. Strom. Placement strategies for internet-scale data stream systems. *Internet Computing, IEEE*, 2008.

[8] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In *ICDE '06*, page 49, 2006.

[9] P. Pietzuch, J. Ledlie, J. Shneidman, M. Welsh, M. Roussoupolos, and M. Seltzer. Evaluating DHT-Based Service Placement for Stream-Based Overlays. Proc. of IPTPS'05, May 2005.

[10] Tanin, Egemen and Nayar, Deepa and Samet, Hanan. An efficient nearest neighbor algorithm for P2P settings. In *Proceedings of the 2005 National Conference on Digital Government Research*, pages 21–28, 2005.

[11] H. W. H. Zvi Drezner. The Fermat-Weber Problem. In *Facility Location: Applications and Theory*, pages 1–24. 2005.