Resilient Blocks for Summarising Distributed Data

Giorgio Audrito
University of Torino, Italy
giorgio.audrito@unito.it

Sergio Bergamini
University of Torino, Italy
sergio.bergamini@edu.unito.it

Summarising distributed data is a central routine for parallel programming, lying at the core of widely used frameworks such as the *map/reduce* paradigm. In the IoT context it is even more crucial, being a privileged mean to allow long-range interactions: in fact, summarising is needed to avoid data explosion in each computational unit.

We introduce a new algorithm for dynamic summarising of distributed data, *weighted multi-path*, improving over the state-of-the-art *multi-path* algorithm. We validate the new algorithm in an archetypal scenario, taking into account sources of volatility of many sorts and comparing it to other existing implementations. We thus show that *weighted multi-path* retains adequate accuracy even in high-variability scenarios where the other algorithms are diverging significantly from the correct values.

1 Introduction

The modern world is increasingly permeated by heterogeneous connected devices, establishing a fully-integrated digital and physical ecosystem. Such a pervasive landscape calls for the adoption of self-organising mechanisms, for which it is challenging and critical to translate the system global requirements into single-agent behaviors. In this context, each agent has a limited knowledge of its environment, which has to be shared with neighbors in order to allow for coordination mechanisms and create a global spatio-temporal representation of the distributed state of computation. Several tools address this problem [3], showing how self-organization can be achieved using a small set of "basic" components, upon which increasingly complex algorithms and behaviors are built [4]. Due to the dynamic scenarios typical of pervasive computing, these components need to carefully trade-off efficiency with resilience to network changes.

The *collection* building block (C in short) is one of the most basic and widely used components, which aggregates values held by devices in a spatial region into a single resulting value in a selected device. This procedure closely resembles the *reduce* phase of the MapReduce paradigm [5], ported into a spatial computing context. This block can be applied to a variety of different contexts, as it can be instantiated for values of any data type with an associative and commutative aggregation operator. However, existing implementations of C (single-path and multi-path [8]) exhibit poor performance whenever node mobility is significant with respect to the device update rate.

In this paper we introduce a new implementation for the C building block, called *weighted multi-path*, which is able to achieve adequate accuracy in highly volatile scenarios. Its performance is then validated in a archetypal situation, taking into account node mobility, update rate variability and discontinuities in network configuration. Section 2 presents the state-of-the-art implementations of C; Section 3 describes the new weighted multi-path algorithm; Section 4 compares the new algorithm with other existing implementations in a archetypal scenario; Section 5 summarises the contributions of this paper and outlines possible future works for improving the weighted multi-path C block.

D. Pianini and G. Salvaneschi (Eds.): First workshop
on Architectures, Languages and Paradigms for IoT
EPTCS 264, 2018, pp. 23–26, doi:10.4204/EPTCS.264.3

2 The Collection Block

The C building block aggregates values held by different devices through a spatial region into a single value in a selected device, by repeated application of an associative and commutative aggregation operator, where each iteration is executed in asynchronous rounds by every device in the network: partially aggregated values flow towards the selected source, while avoiding multiple aggregation of the same values. This two-faceted prerequisite, of *acyclic* flows *directed* towards the source, is met by relying on a given *potential field*, approximating a certain measure of distance from the selected source. As long as information flows descending the potential field, cyclic dependencies are prevented and eventual reaching of the source is guaranteed. The C block thus computes a "summary" of given values v_δ in the local minima of a potential field $P(\delta)$, through a binary aggregating operator $\oplus$. Potential descent is enforced by splitting neighborhoods according to their potential value, obtaining the two disjoint sets:

$$D_\delta^- = \left\{\delta' \text{ linked to } \delta \text{ such that } P(\delta') < P(\delta)\right\}$$
$$D_\delta^+ = \left\{\delta' \text{ linked to } \delta \text{ such that } P(\delta') > P(\delta)\right\}$$

Two different implementations of the collection block have been proposed so far: *single-path* and *multi-path*. The single-path strategy C_{sp} ensures that information flows through a forest in the network, by sending the whole partial aggregate of a device δ to the single device $m(\delta)$ with *minimal* potential among devices connected to δ. This is accomplished by repeatedly applying the following update rule:

$$C_{\text{sp}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D_\delta^+ \wedge m(\delta') = \delta} C_{\text{sp}}(\delta'),$$

which computes the partial aggregate in δ by combining together the value in δ and the partial aggregates in devices with higher potential for which δ is the selected output device $m(\delta')$.

The multi-path strategy C_{mp}, instead, allows information to flow through every path compatible with the given potential field. The partial aggregate of a device δ is thus divided equally among *every* device δ' connected to δ with lower potential, by iteratively applying the following update rule:

$$C_{\text{mp}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D_\delta^+} \left\{C_{\text{mp}}(\delta') \oslash |D_{\delta'}^-|\right\};$$

where $\oslash$ is a binary operator $v \oslash n$ "extracting the n-th root", i.e., an element which aggregated with itself n times produces the original value v. Since information needs to be "divisible" for $\oslash$ to exist, *multi-path* has a narrower scope than *single-path*. However, both arithmetic-like operations $(+, \times, \ldots)$ and idempotent operations $(\min, \max, \ldots)$ are *divisible* (through resp. division, root extraction, identity); so that the situations where *multi-path* is not applicable are rare in practice.

3 Weighted Multi-path C

The *weighted multi-path* C develops on the multi-path strategy, by allowing partial aggregates to be divided unequally among neighbors. *Weights* corresponding to neighbors are calculated in order to penalize devices which are likely to lose their "receiving" status, situation which can happen in two cases:

1. if the "receiving" device is too close to the edge of the communication range of the "sending" device, it might step outside of it in the immediate future breaking the connection;
2. if the potential of the "receiving" device is too close to the potential of the "sending" device, their relative role of sender/receiver might be switched in the immediate future, possibly creating an "information loop" between the two devices.

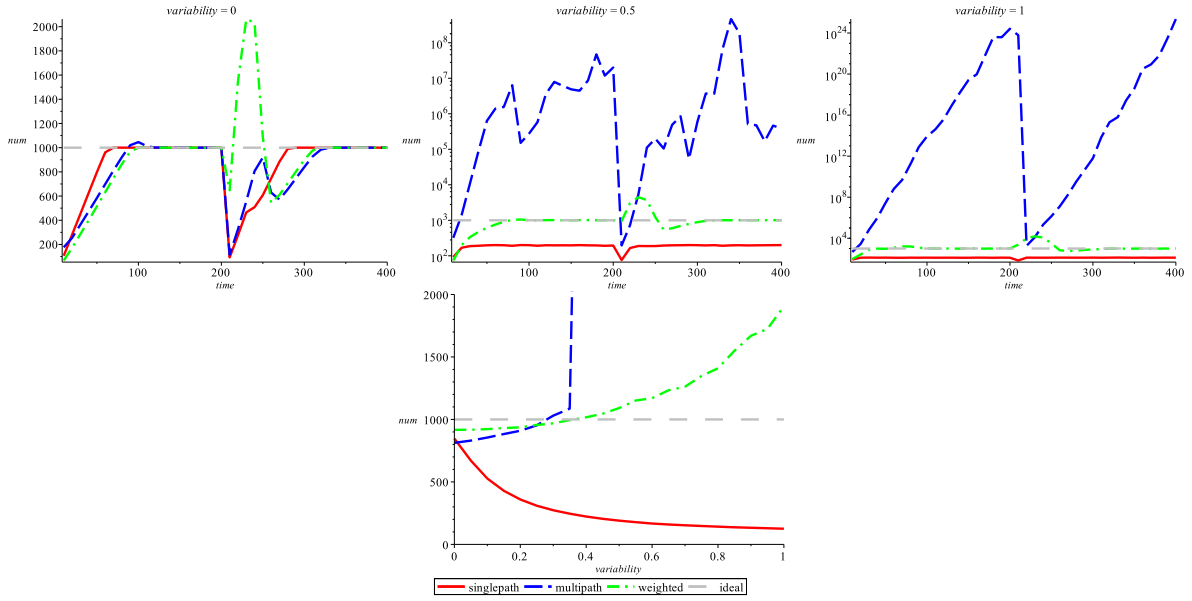

Figure 1: Number of devices measured through implementations of C, upon increasing variability with source change at $T = 200s$. Evolution through time is presented for variabilities 0, 0.5, 1 (top), together with the average number in time window $0-400s$ for 21 variabilities from 0 to 1 (bottom).

We can address both of these situations with a natural weight function $w(\delta,\delta')$, measuring how much of the information from δ should flow to δ', as the product of two corresponding factors $d(\delta,\delta')\cdot p(\delta,\delta')$:

$$d(\delta,\delta') = R - D(\delta,\delta'), \qquad p(\delta,\delta') = \left|P(\delta) - P(\delta')\right|$$

where R is the communication radius, $D(\delta,\delta')$ is the physical distance between devices δ,δ' and $P(\delta)$ is the potential of device δ. Notice that w is positive and symmetric, hence it can be interpreted as an attribute of connection links representing both the amount of information to "send" and to "receive". Since the $w(\delta,\delta')$ values do not sum up to any particular value, they need to be normalized by the factor $N(\delta) = \sum_{\delta'\in D_\delta^-} w(\delta,\delta')$ before use, obtaining normalized weights $w(\delta,\delta')/N(\delta')$. The partial aggregates accumulated by devices can then be calculated as in C_{mp} with the addition of weights, by iteratively applying the following update rule:

$$C_{\text{wmp}}(\delta) = v_\delta \oplus \bigoplus_{\delta'\in D_\delta^+} \left\{ C_{\text{wmp}}(\delta') \otimes \frac{w(\delta',\delta)}{N(\delta')} \right\};$$

where $\otimes$ is a binary operator $v\otimes k$ "extracting" a certain percentage k of a local value v.

4 Evaluation

We compared *weighted*, *multi-path* and *single-path* in an archetypal scenario of 1000 devices randomly distributed along a $200m \times 20m$ corridor, with a $1s$ average computation rate and a $10m$ communication range. The source device was located on the right end of the corridor, then discontinuously moved to the left end at time $T = 200s$. The potential field was computed through BIS gradient [2], and the aggregation chosen to count the total number of devices (thus setting $\oplus = +$, $\otimes = \times$ and $v = 1$ for each device). We

tested degrees of variability ranging from 0 (no movement, regular computation rate) to 1 (short- and long-range movements, irregular computation rate in each device and between different devices), thus testing the algorithm in a scenario of increasing variability aimed at reproducing the worst possible case.

Figure 1 summarises the evaluation results; obtained with Protelis [7] as programming language, Alchemist as simulator [6] and the supercomputer OCCAM [1] as platform[1]. We run 100 instances of each scenario and averaged the results, which had a significant relative standard error between them for high values of variability. The tests showed that *single-path* systematically underestimates the ideal value, with a similarly poor performance under variability 0.5 and 1 showing that accurate values are attainable only for low-variability scenarios. Conversely, *multi-path* overestimates the value with an exponentially-growing behavior (randomly "reset" from time to time) which gets exponentially worse with increased variability. On the other hand, *weighted multi-path* achieves an adequate accuracy even in highly volatile scenarios, scoring the best results under steady inputs for every value of variability. The recovery speed from input discontinuities is also comparable with that of the other algorithms, however, the resulting transients contain value peaks that are sometimes higher than those of other algorithms.

5 Conclusion

We introduced *weighted multi-path C*, a new algorithm for summarising distributed data improving over state-of-the-art implementations of the C block. Experimental evaluation shows that *weighted* achieves adequate accuracy even in high-variability scenarios where other approaches are infeasible. However, the new algorithm suffers from high (though short) error peaks in response to input discontinuities: in these situations, potential-descending data flows create loops leading to exponential increases in error. Future works could address this problem, either through detecting input discontinuities for triggering corrective actions, or by preventing the creation of loops through time-driven potential fields.

References

[1] M. Aldinucci, S. Bagnasco, S. Lusso, P. Pasteris, S. Vallero & S. Rabellino (2016): *The Open Computing Cluster for Advanced data Manipulation (OCCAM)*. In: *The 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP)*, San Francisco, USA.

[2] G. Audrito, F. Damiani & M. Viroli (2017): *Optimally-Self-Healing Distributed Gradient Structures Through Bounded Information Speed*. In: *Coordination Languages and Models*, *LNCS* 10319, Springer-Verlag, doi:10.1007/978-3-319-59746-1_4.

[3] J. Beal, S. Dulman, K. Usbeck, M. Viroli & N. Correll (2013): *Organizing the Aggregate: Languages for Spatial Computing*. In: *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*, chapter 16, IGI Global, pp. 436–501, doi:10.4018/978-1-4666-2092-6.ch016.

[4] J. Beal, D. Pianini & M. Viroli (2015): *Aggregate Programming for the Internet of Things*. *IEEE Computer*, doi:10.1109/MC.2015.261.

[5] J. Dean & S. Ghemawat (2008): *MapReduce: simplified data processing on large clusters*. *Commun. ACM* 51(1), pp. 107–113, doi:10.1145/1327452.1327492.

[6] D. Pianini, S. Montagna & M. Viroli (2013): *Chemical-oriented simulation of computational systems with ALCHEMIST*. *J. Simulation* 7(3), pp. 202–215, doi:10.1057/jos.2012.27.

[7] D. Pianini, M. Viroli & J. Beal (2015): *Protelis: Practical Aggregate Programming*. In: *ACM SAC 2015*, doi:10.1145/2695664.2695913.

[8] M. Viroli, J. Beal, F. Damiani & D. Pianini (2015): *Efficient Engineering of Complex Self-Organising Systems by Self-Stabilising Fields*. In: *IEEE SASO 2015*, IEEE, pp. 81–90, doi:10.1109/SASO.2015.16.

[1] For the sake of reproducibility, the actual experiment is made available at `https://bitbucket.org/gaudrito/audrito-bergamini-summarising`