@Article{Leiserson_2020,

```
author    = {Charles E. Leiserson and Neil C. Thompson and Joel S. Emer and
Bradley C. Kuszmaul and Butler W. Lampson and Daniel Sanchez and Tao B.
Schardl},
journal   = {Science},
loc       = {Science},
title     = {There's plenty of room at the Top: What will drive computer
performance after Moore's law?},
year      = {2020},
month     = {jun},
number    = {6495},
pages     = {eaam9744},
volume    = {368},
doi       = {10.1126/science.aam9744},
publisher = {American Association for the Advancement of Science ({AAAS})},
url       =
{https://www.microsoft.com/en-us/research/uploads/prod/2020/11/Leiserson-et-
al-Theres-plenty-of-room-at-the-top.pdf}
```

}

@inbook{10.1145/3453483.3454079,

```
author    = {Morihata, Akimasa and Sato, Shigeyuki},
title     = {Reverse Engineering for Reduction Parallelization via Semiring
Polynomials},
year      = {2021},
isbn      = {9781450383912},
publisher = {Association for Computing Machinery},
address   = {New York, NY, USA},
url       = {https://doi.org/10.1145/3453483.3454079},
abstract  = {Parallel reduction, which summarizes a given dataset, e.g., the
total, average, and maximum, plays a crucial role in parallel programming.
This paper presents a new approach, reverse engineering, to automatically
discovering nontrivial parallel reductions in sequential programs. The body
of the sequential reduction loop is regarded as a black box, and its input-
output behaviors are sampled. If the behaviors correspond to a set of linear
polynomials over a semiring, a divide-and-conquer parallel reduction is
generated. Auxiliary reverse-engineering methods enable a long and nested
loop body to be decomposed, which makes our parallelization scheme
applicable to various types of reduction loops. This approach is not only
simple and efficient but also agnostic to the details of the input program.
Its potential is demonstrated through several use case scenarios. A proof-
of-concept implementation successfully inferred linear polynomials for
nearly all of the 74 benchmarks exhaustively collected from the literature.
These characteristics and experimental results demonstrate the promise of
the proposed approach, despite its inherent unsoundness.},
booktitle = {Proceedings of the 42nd ACM SIGPLAN International Conference on
Programming Language Design and Implementation},
loc       = {Proceedings of the 42nd ACM SIGPLAN International Conference on
```

```
Programming Language Design and Implementation},
number    = {2021},
pages     = {820–834},
numpages  = {15}
```

}

@inproceedings{10.1145/3243176.3243204,

```
author    = {Jiang, Peng and Chen, Linchuan and Agrawal, Gagan},
title     = {Revealing Parallel Scans and Reductions in Recurrences through
Function Reconstruction},
year      = {2018},
isbn      = {9781450359863},
publisher = {Association for Computing Machinery},
address   = {New York, NY, USA},
url       = {https://doi.org/10.1145/3243176.3243204},
doi       = {10.1145/3243176.3243204},
abstract  = {Many sequential loops are actually recurrences and can be
parallelized across iterations as scans or reductions. Many efforts over the
past 2+ decades have focused on parallelizing such loops by extracting and
exploiting the hidden scan/reduction patterns. These approaches have largely
been based on a heuristic search for closed-form composition of computations
across loop iterations.While the search-based approaches are successful in
parallelizing many recurrences, they have a large search overhead and need
extensive program analysis. In this work, we propose a novel approach called
sampling-and-reconstruction, which avoids the search for closed-form
composition and has the potential to cover more recurrence loops. It is
based on an observation that many recurrences can have a point-value
representation. The loop iterations are divided across processors, and where
the initial value(s) of the recurrence variable(s) are unknown, we execute
with several chosen (sampling) initial values. Then, correct final result
can be obtained by reconstructing the function from the outputs produced on
the chosen initial values. Our approach is effective in parallelizing
linear, rectified-linear, finite-state and multivariate recurrences, which
cover all of the test cases in previous works. Our evaluation shows that our
approach can parallelize a diverse set of sequential loops, including cases
that cannot be parallelized by a state-of-the-art static parallelization
tool, and achieves linear scalability across multiple cores.},
booktitle = {Proceedings of the 27th International Conference on Parallel
Architectures and Compilation Techniques},
loc       = {Proceedings of the 27th International Conference on Parallel
Architectures and Compilation Techniques},
number    = {2018},
articleno = {10},
numpages  = {13},
keywords  = {loop parallelization, recurrence, reduction},
location  = {Limassol, Cyprus},
series    = {PACT '18}
```

}

@misc{blleloch2019improved,

```
title         = {Improved Parallel Cache-Oblivious Algorithms for Dynamic
Programming and Linear Algebra},
author        = {Guy E. Blleloch and Yan Gu},
year          = {2019},
eprint        = {1809.09330},
archivePrefix = {arXiv},
primaryClass  = {cs.DS},
loc           = {arXiv},
number        = {1809.09330},
url           = {https://arxiv.org/abs/1809.09330}
```

}

@inproceedings{Henry_2021,

```
title         = {Compilation of Sparse Array Programming Models},
author        = {Rawn Henry, Olivia Hsu, Rohan Yadav, Stephen Chou, Kunle
Olukotun, Saman Amarasinghe, and Fredrik
```

Kjolstad},

```
year          = {2021},
articleno     = {128},
numpages      = {29},
url           =
{http://fredrikbk.com/publications/Sparse_Array_Programming.pdf},
publisher     = {Association for Computing Machinery},
loc           = {Proc. ACM Program. Lang. 5},
number        = {},
doi           = {10.1145/3485505}
```

}

@InProceedings{10.1007/3-540-17179-7_30,

```
author       = {Rajopadhye, Sanjay V. and Purushothaman, S. and Fujimoto,
Richard M.},
editor       = {Nori, Kesav V.},
title        = {On synthesizing systolic arrays from Recurrence Equations
with Linear Dependencies},
booktitle    = {Foundations of Software Technology and Theoretical Computer
Science},
year         = {1986},
publisher    = {Springer Berlin Heidelberg},
address      = {Berlin, Heidelberg},
pages        = {488--503},
abstract     = {We present a technique for synthesizing systolic
```

```
architectures from Recurrence Equations. A class of such equations
(Recurrence Equations with Linear Dependencies) is defined and the problem
of mapping such equations onto a two dimensional architecture is studied. We
show that such a mapping is provided by means of a linear allocation and
timing function. An important result is that under such a mapping the
dependencies remain linear. After obtaining a two-dimensional architecture
by applying such a mapping, a systolic array can be derived if the
communication can be spatially and temporally localized. We show that a
simple test consisting of finding the zeroes of a matrix is sufficient to
determine whether this localization can be achieved by pipelining and give a
construction that generates the array when such a pipelining is possible.
The technique is illustrated by automatically deriving a well known systolic
array for factoring a band matrix into lower and upper triangular factors.},
isbn         = {978-3-540-47239-1},
loc          = {Foundations of Software Technology and Theoretical Computer
Science},
number       = {},
doi          = {10.1007/3-540-17179-7_30},
url          = {https://link.springer.com/chapter/10.1007/3-540-17179-7_30}
```

}