

Cleveland State University  
**EngagedScholarship@CSU**

Undergraduate Research Posters 2015

Undergraduate Research Posters

2015

# Parallel Selection Algorithms on GPUs: Implementation and Performance Comparison

Darius Bakunas-Milanowski  
*Cleveland State University*

Follow this and additional works at: [https://engagedscholarship.csuohio.edu/u\\_poster\\_2015](https://engagedscholarship.csuohio.edu/u_poster_2015)

**How does access to this work benefit you? Let us know!**

## Recommended Citation

Bakunas-Milanowski, Darius, "Parallel Selection Algorithms on GPUs: Implementation and Performance Comparison" (2015).  
*Undergraduate Research Posters 2015*. 58.  
[https://engagedscholarship.csuohio.edu/u\\_poster\\_2015/58](https://engagedscholarship.csuohio.edu/u_poster_2015/58)

This Book is brought to you for free and open access by the Undergraduate Research Posters at EngagedScholarship@CSU. It has been accepted for inclusion in Undergraduate Research Posters 2015 by an authorized administrator of EngagedScholarship@CSU. For more information, please contact [library.es@csuohio.edu](mailto:library.es@csuohio.edu).



This digital edition was prepared by MSL Academic Endeavors, the imprint of the Michael Schwartz Library at Cleveland State University.

# ***Parallel Selection Algorithms on GPUs: Implementation and Performance Comparison***

Washkewicz College of Engineering

**Student Researcher:** Darius Bakunas-Milanowski

**Faculty Advisor:** Janche Sang

## **Abstract**

The computing power of current Graphical Processing Units (GPUs) has increased rapidly over the years. They offer much more computational power than recent CPUs by providing a vast number of simple, data parallel, multithreaded cores. In this project, we focused on the study of different variations of parallel selection algorithms on the current generation of NVIDIA GPUs. That is, given a massively large array of elements, we were interested in how we could use a GPU to efficiently select those elements that meet certain criteria and then store them into a target array for further processing. The optimization techniques used and implementation issues encountered are discussed in detail. Furthermore, the experiment results show that our advanced implementation performs an average of 1.74 times faster than Thrust, an open-source parallel algorithms library.