```
\documentclass{article}

\usepackage[utf8]{inputenc}

\usepackage{enumerate}

\usepackage{graphics}

\usepackage{graphicx}


\title{Lab 6}

\author{Jack Jiang}

\date{April 2, 2021}


\begin{document}


\maketitle


\section*{Analysis}

\begin{enumerate}[(a)]

    \item In scatter\_Vector, I analyzed the code and wrote it out on a paper to fully understand what was going on. From the code I analyzed, I was led to believe that to begin, the total vector size had to be shared amongst the other ranks so that occurred. When rank was 0, the variable containing the total vector size was sent to the other ranks using MPI\_send and when rank was not 0, the total vector size was received using MPI\_Recv. In the next part, the vector was scattered and so the vector size of a local vector was obtained by dividing the total vector size by the number of processors and the last rank also had the remainder added to its vector size. Next, if the rank was 0, its v\_local vector was created and the original vector v had its data sent to the other ranks accordingly (based on index/size of v\_local), where the other ranks received the data.

    \item \includegraphics[strong]{StrongAxpy.png}

    \includegraphics[weak]{WeakAxpy.png}

    I do not see good strong or weak scalability because I cannot tell due to the 2 plots looking the same. This may be due to me incorrectly executing strong and weak scalability studies.

    \item We do not see ideal speedup in this code because even if distributed axpy might be embarrassingly parallel mathematically, it is a simple operation and so speedup is not efficiently maximized in this program.

    \item I helped Shaan Chudasama with MPI and sbatch concepts.
```

```
\end{enumerate}

\end{document}
```