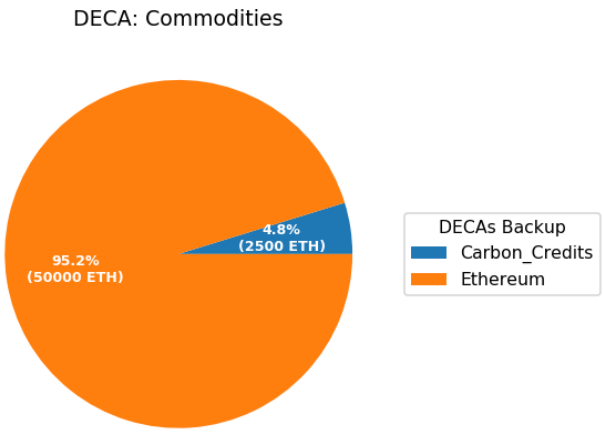
In [1]:
```
#%matplotlib inline
%matplotlib notebook
import numpy as np

#ethereum price USD
ethPrice = 200
#ccPercentage
ccPercentage = 5
#Total Investment USD
totalInvestment = 10000000

# Carbon Credits Price Sum in ETH
ccPriceSum = (totalInvestment * ccPercentage) / (100 * ethPrice)
print("Carbon Credits Total(ETH):{}".format(ccPriceSum))

#Ethereum Price Sum in ETH
ethPriceSum = totalInvestment / ethPrice
print("Total Ethereums:{}".format(ethPriceSum))

#Consider generating random numbers between that match promotios per ethereum that we got
to reach our goal:
promo = [300,275,250,225]
DTSlst = np.random.randint(0, 4, int(ethPriceSum))
DTSlst = np.array(list(map(lambda x: promo[x], DTSlst)))

#total supply per ethereums + decas to contract owner
DTS = 1.025* DTSlst.sum()
print("DECAS Total Supply:{}".format(DTS))

#Price Per Deca in ETH
PPD = (ccPriceSum + ethPriceSum) / DTS
print("Price per Deca:{}".format(PPD))

#Verification Code:
assert (ethPriceSum+ccPriceSum==PPD*DTS), "Error in verification"
print('{0}={1}'.format(ethPriceSum+ccPriceSum,PPD*DTS))
```

```
Carbon Credits Total(ETH):2500.0
Total Ethereums:50000.0
DECAS Total Supply:13459454.374999998
Price per Deca:0.0039006038831347505
52500.0=52500.0
```

In [2]:
```
import matplotlib.pyplot as plt
#the pie chart
fig, ax = plt.subplots(figsize=(9, 4.5), subplot_kw=dict(aspect="equal"))

commodities = ["{} Carbon_Credits".format(ccPriceSum),
          "{} Ethereum".format(ethPriceSum)]

data = [float(x.split()[0]) for x in commodities]
backup = [x.split()[-1] for x in commodities]


def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}%\n({:d} ETH)".format(pct, absolute)


wedges, texts, autotexts = ax.pie(data, autopct=lambda pct: func(pct, data),
                                  textprops=dict(color="w"))

ax.legend(wedges, backup,
          title="DECAs Backup",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=8, weight="bold")

ax.set_title("DECA: Commodities")

plt.show()
```



In [3]:
```
from IPython.display import display, HTML
from fix42D import fix42D
import pandas as pd

#create the DataFrame
preICOStart = '07-01-2020' #MM-DD-YYYY
days = 77
dRan1 = pd.date_range(start = preICOStart, periods = days)

#77 days dataframe
df = pd.DataFrame(dRan1)
df = df.rename(columns={0: "dates"})
#df = df.set_index('dates', drop = True)

#Total DECAS by Day (bought)
df1 = pd.DataFrame(fix42D(days,DTSlst))
df1 = df1.sum(axis=1)
df1 = df1.to_frame(name = 'TotalDecasByDay')

#Total Ethereums by Day (invested)
ETHlst = np.ones(int(ethPriceSum))
df2 = pd.DataFrame(fix42D(days,ETHlst))
df2 = df2.sum(axis=1)
df2 = df2.to_frame(name = 'TotalETHByDay')

#Ethereum Prices by Day (this must fluctuate in real life)
#just a list of ethereum price example (useful for charts)
priceETHlst = np.ones(days).reshape(-1,1)
priceETHlst = priceETHlst * ethPrice
df3 = pd.DataFrame.from_records(priceETHlst)
df3.columns = ['PriceETHByDay(USD)']

#concat the dataframes
result = pd.concat([df, df1, df2, df3], axis=1, sort=False)
#Deca Total Supply by day
result['DecaTotalSupply'] = 0
for i in range(days):
    result.loc[i,'DecaTotalSupply'] = result.loc[:i,'TotalDecasByDay'].sum() * 1.025
#Ethereum Total Supply by day
result['EthereumTotalSupply'] = 0
for i in range(days):
    result.loc[i,'EthereumTotalSupply'] = result.loc[:i,'TotalETHByDay'].sum()
#carbon Credits backup
result['ccPriceSum(ETH)'] = ccPriceSum
#Price Per Deca in Ethereum
# based on price formula  PPD(ETH) = (ccPriceSum + ethPriceSum) / DTS
result['DECAPrice(ETH)'] = 0
for i in range(days):
    result.loc[i,'DECAPrice(ETH)'] = (result.loc[i,'ccPriceSum(ETH)']
                                      + result.loc[i,'EthereumTotalSupply']) / result.loc[i,'D
ecaTotalSupply']
# LOWEST Price Per Deca
# based on price formula  PPD = (ccPriceSum) / DTS
result['DECALowestPrice(ETH)'] = 0
for i in range(days):
    result.loc[i,'DECALowestPrice(ETH)'] = (result.loc[i,'ccPriceSum(ETH)']) / result.loc
[i,'DecaTotalSupply']

# Percentage Backup
# based on PBCK = ( DECALowestPrice / DECAPrice) * 100
result['PercentageBackUp'] = 0
for i in range(days):
    result.loc[i,'PercentageBackUp'] = (result.loc[i,'DECALowestPrice(ETH)'] / result.loc
[i,'DECAPrice(ETH)']) * 100

display(result)
result.to_csv("result.csv")
```

	dates	TotalDecasByDay	TotalETHByDay	PriceETHByDay(USD)	DecaTotalSupply	EthereumTotalSupply	ccPriceSu
0	2020-07-01	170175	650.0	200.0	1.744294e+05	650.0	
1	2020-07-02	170950	650.0	200.0	3.496531e+05	1300.0	
2	2020-07-03	171050	650.0	200.0	5.249794e+05	1950.0	
3	2020-07-04	171175	650.0	200.0	7.004337e+05	2600.0	
4	2020-07-05	171550	650.0	200.0	8.762725e+05	3250.0	
...	...	...	...	...	...	...	
72	2020-09-11	171600	650.0	200.0	1.277229e+07	47450.0	
73	2020-09-12	171300	650.0	200.0	1.294788e+07	48100.0	
74	2020-09-13	170875	650.0	200.0	1.312302e+07	48750.0	
75	2020-09-14	170675	650.0	200.0	1.329797e+07	49400.0	
76	2020-09-15	157550	600.0	200.0	1.345945e+07	50000.0	

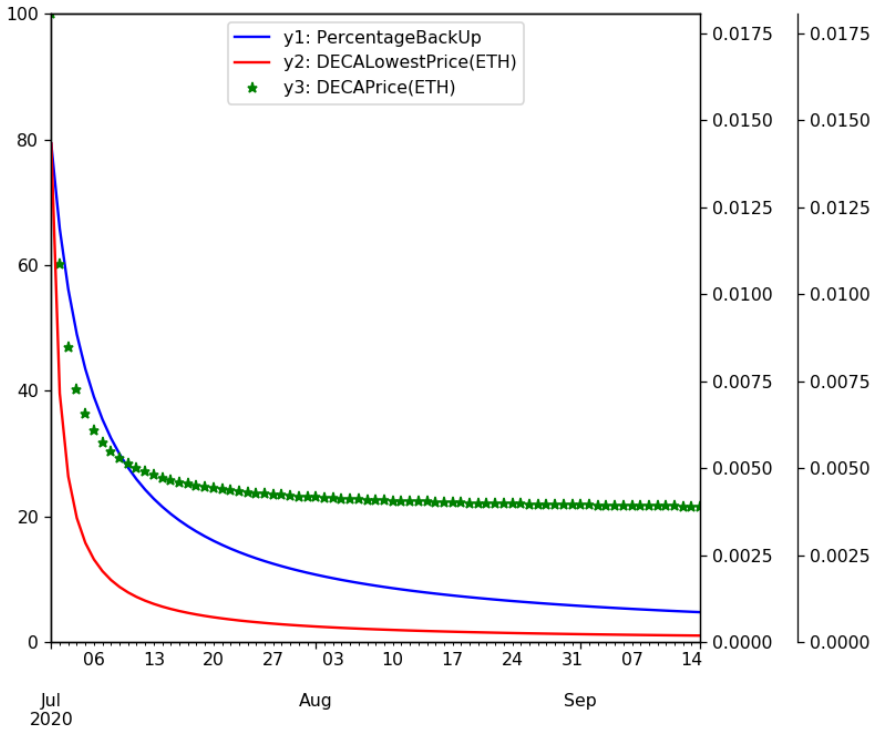
77 rows × 10 columns

In [4]:

```
#plot Percentage Backup
#get the important cols
finalDF1 = result[['dates','PercentageBackUp', 'DECALowestPrice(ETH)', 'DECAPrice(ETH)']]
finalDF1 = finalDF1.set_index('dates', drop = True)
display(finalDF1)
#to excel
finalDF1.to_csv("PercentageBackUp.csv")
#prepare chart

fig, ax = plt.subplots(figsize=(9, 6.5))
ax3 = ax.twinx()
rspine = ax3.spines['right']
rspine.set_position(('axes', 1.15))
ax3.set_frame_on(True)
ax3.patch.set_visible(False)
fig.subplots_adjust(right=0.7)

finalDF1['PercentageBackUp'].plot(ax=ax, style='b-')
# same ax as above since it's automatically added on the right
finalDF1['DECALowestPrice(ETH)'].plot(ax=ax, style='r-', secondary_y=True)
finalDF1['DECAPrice(ETH)'].plot(ax=ax3, style='g*')

#Set Y Axis Limits
ax.set_ylim([0,100]) #Percentage backup
ax3.set_ylim([0,finalDF1['DECAPrice(ETH)'].max()]) #DECAPrice
ax.right_ax.set_ylim([0,finalDF1['DECAPrice(ETH)'].max()]) #DecaLowestPrice

# add legend --> take advantage of pandas providing us access
# to the line associated with the right part of the axis
ax3.legend([ax.get_lines()[0], ax.right_ax.get_lines()[0], ax3.get_lines()[0]],\
           ['y1: PercentageBackUp','y2: DECALowestPrice(ETH)','y3: DECAPrice(ETH)'],loc =
'upper center', bbox_to_anchor=(0.5, 1.0))
#plt.savefig("DecaMarketLowest.png")
#finalDF1.plot()
```

dates	PercentageBackUp	DECALowestPrice(ETH)	DECAPrice(ETH)
2020-07-01	79.365079	0.014332	0.018059
2020-07-02	65.789474	0.007150	0.010868
2020-07-03	56.179775	0.004762	0.008477
2020-07-04	49.019608	0.003569	0.007281
2020-07-05	43.478261	0.002853	0.006562
...	...	...	...
2020-09-11	5.005005	0.000196	0.003911
2020-09-12	4.940711	0.000193	0.003908
2020-09-13	4.878049	0.000191	0.003905
2020-09-14	4.816956	0.000188	0.003903
2020-09-15	4.761905	0.000186	0.003901

77 rows × 3 columns



Out[4]: <matplotlib.legend.Legend at 0x7f8fb65a6cd0>

In [5]:
```
### Extension proposal for next 3 years
### 3 times 365 +1
daysToAdd = 1096

#create the DataFrame Days
#get the last day
ICOEnds = dRan1[[-1]].strftime("%m-%d-%Y")
dRan2 = pd.date_range(start = ICOEnds[0], periods = daysToAdd)
#remove first day which is preICOEnds
dRan2 = dRan2[1:]
df4 = pd.DataFrame(dRan2)
df4 = df4.rename(columns={0: "dates"})

#AfterICOStartDate first day
AfterICOStarts = dRan2[[0]].strftime("%m-%d-%Y")

#take the last Deca Total Supply since there won't be more
df4['DecaTotalSupply'] = result.loc[len(result) -1 ,'DecaTotalSupply']
#take the last Ethereum Total Supply since there won't be more
df4['EthereumTotalSupply'] = result.loc[len(result) -1 ,'EthereumTotalSupply']
#define dates and % backup and EthereumPrice list for buying and canceling carbon credits
dates = [[AfterICOStarts,5 ,ethPrice],
    ['09-15-2021', 10, ethPrice],
    ['09-15-2022', 15, ethPrice ],
    ['09-15-2023', 20, ethPrice ]]

'''
    we write multiple whiles for each column calculation to make it easier to understand.
'''
#while for Ethereum prices, supose it moves as dates in the third column:
df4['ethPrice'] = ethPrice
#each day
i = 0
#each date from dates list
date = 0
while i < (daysToAdd-1) and date <= len(dates[0]):
    #get the current day
    currentDay = df4.loc[i,'dates'].strftime("%m-%d-%Y")
    #if the day is a define date, add the price and change to next date data
    if currentDay == dates[date][0]:
        df4.loc[i,'ethPrice'] = dates[date][2]
        date+=1
        i+=1
    #add ethereum proposed prices when buying C
    else:
        df4.loc[i,'ethPrice'] = dates[date-1][2]
        i+=1
#import pdb; pdb.set_trace()
#while for getting percentages as bought and cc cancelation:
df4['ccPriceSum(ETH)'] = ccPriceSum
#percent variable for debug
#df4['percent'] = 0
#each day
i = 0
#each date from dates list
date = 0
while i < (daysToAdd-1) and date <= len(dates[0]):
    #get the current day
    currentDay = df4.loc[i,'dates'].strftime("%m-%d-%Y")
    #if the day is a define date, add the price and change to next date data
    if currentDay == dates[date][0]:
        #first we buy and cancel carbon credits with the current Ethereum Price
        df4.loc[i,'ccPriceSum(ETH)'] +=  df4.loc[i,'EthereumTotalSupply'] * (dates[dat
e][1]/100)
        #then we take out the percentage from eth and update the eth supply
        df4.loc[i,'EthereumTotalSupply'] = df4.loc[i,'EthereumTotalSupply'] * (1 - dates
[date][1]/100)
        #df4.loc[i,'percent'] = dates[date][1]
        date+=1
```

	dates	DecaTotalSupply	EthereumTotalSupply	ethPrice	ccPriceSum(ETH)	DECAPrice(ETH)	DECALowestPrice(E
0	2020-09-16	1.345945e+07	47500.0	200	5000.0	0.003901	0.000
1	2020-09-17	1.345945e+07	45000.0	200	7500.0	0.003901	0.000
2	2020-09-18	1.345945e+07	45000.0	200	7500.0	0.003901	0.000
3	2020-09-19	1.345945e+07	45000.0	200	7500.0	0.003901	0.000
4	2020-09-20	1.345945e+07	45000.0	200	7500.0	0.003901	0.000
...	...	...	...	...	...	...	
1090	2023-09-11	1.345945e+07	40000.0	200	12500.0	0.003901	0.000
1091	2023-09-12	1.345945e+07	40000.0	200	12500.0	0.003901	0.000
1092	2023-09-13	1.345945e+07	40000.0	200	12500.0	0.003901	0.000
1093	2023-09-14	1.345945e+07	40000.0	200	12500.0	0.003901	0.000
1094	2023-09-15	1.345945e+07	40000.0	200	12500.0	0.003901	0.000

1095 rows × 8 columns

In [6]:
```
resultICO = result[['dates','DecaTotalSupply','EthereumTotalSupply','PriceETHByDay(USD)','ccPriceSum(ETH)','DECAPrice(ETH)','DECALowestPrice(ETH)','PercentageBackUp']]
#rename PriceETHByDay to ethPrice
resultICO.rename(columns={'PriceETHByDay(USD)':'ethPrice'}, inplace=True)
resultFull = pd.concat([resultICO, df4], axis=0, sort=False)
#resultFull = resultFull.set_index('dates', drop = True)
display(resultFull)
```

```
/home/p1r0/Dev/labs/DECAJupyter/myenv/lib/python3.7/site-packages/pandas/core/frame.py:4133: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

	dates	DecaTotalSupply	EthereumTotalSupply	ethPrice	ccPriceSum(ETH)	DECAPrice(ETH)	DECALowestPrice(E
0	2020-07-01	1.744294e+05	650.0	200.0	2500.0	0.018059	0.014
1	2020-07-02	3.496531e+05	1300.0	200.0	2500.0	0.010868	0.007
2	2020-07-03	5.249794e+05	1950.0	200.0	2500.0	0.008477	0.004
3	2020-07-04	7.004337e+05	2600.0	200.0	2500.0	0.007281	0.003
4	2020-07-05	8.762725e+05	3250.0	200.0	2500.0	0.006562	0.002
...	...	...	...	...	...	...	
1090	2023-09-11	1.345945e+07	40000.0	200.0	12500.0	0.003901	0.000
1091	2023-09-12	1.345945e+07	40000.0	200.0	12500.0	0.003901	0.000
1092	2023-09-13	1.345945e+07	40000.0	200.0	12500.0	0.003901	0.000
1093	2023-09-14	1.345945e+07	40000.0	200.0	12500.0	0.003901	0.000
1094	2023-09-15	1.345945e+07	40000.0	200.0	12500.0	0.003901	0.000

1172 rows × 8 columns

In [7]:
```
#plot Percentage Backup
#get the important cols
finalDF2 = resultFull[['dates','PercentageBackUp', 'DECALowestPrice(ETH)', 'DECAPrice(ETH)']]
finalDF2 = finalDF2.set_index('dates', drop = True)
display(finalDF2)
#to excel
finalDF2.to_csv("PercentageBackUpFinall.csv")
#prepare chart

fig, ax = plt.subplots(figsize=(9, 6.5))
ax3 = ax.twinx()
rspine = ax3.spines['right']
rspine.set_position(('axes', 1.15))
ax3.set_frame_on(True)
ax3.patch.set_visible(False)
fig.subplots_adjust(right=0.7)

finalDF2['PercentageBackUp'].plot(ax=ax, style='b-')
# same ax as above since it's automatically added on the right
finalDF2['DECALowestPrice(ETH)'].plot(ax=ax, style='r-', secondary_y=True)
finalDF2['DECAPrice(ETH)'].plot(ax=ax3, style='g-')

#Set Y Axis Limits
ax.set_ylim([0,100]) #Percentage backup
ax3.set_ylim([0,finalDF2['DECAPrice(ETH)'].max()]) #DECAPrice
ax.right_ax.set_ylim([0,finalDF2['DECAPrice(ETH)'].max()]) #DecaLowestPrice

# add legend --> take advantage of pandas providing us access
# to the line associated with the right part of the axis
ax3.legend([ax.get_lines()[0], ax.right_ax.get_lines()[0], ax3.get_lines()[0]],\
           ['y1: PercentageBackUp','y2: DECALowestPrice(ETH)','y3: DECAPrice(ETH)'],loc = 'upper center', bbox_to_anchor=(0.5, 1.0))
plt.savefig("DecaMarketLowestFull.png")
#finalDF2.plot()
```

dates	PercentageBackUp	DECALowestPrice(ETH)	DECAPrice(ETH)
2020-07-01	79.365079	0.014332	0.018059
2020-07-02	65.789474	0.007150	0.010868
2020-07-03	56.179775	0.004762	0.008477
2020-07-04	49.019608	0.003569	0.007281
2020-07-05	43.478261	0.002853	0.006562
...	...	...	...
2023-09-11	23.809524	0.000929	0.003901
2023-09-12	23.809524	0.000929	0.003901
2023-09-13	23.809524	0.000929	0.003901
2023-09-14	23.809524	0.000929	0.003901
2023-09-15	23.809524	0.000929	0.003901

1172 rows × 3 columns